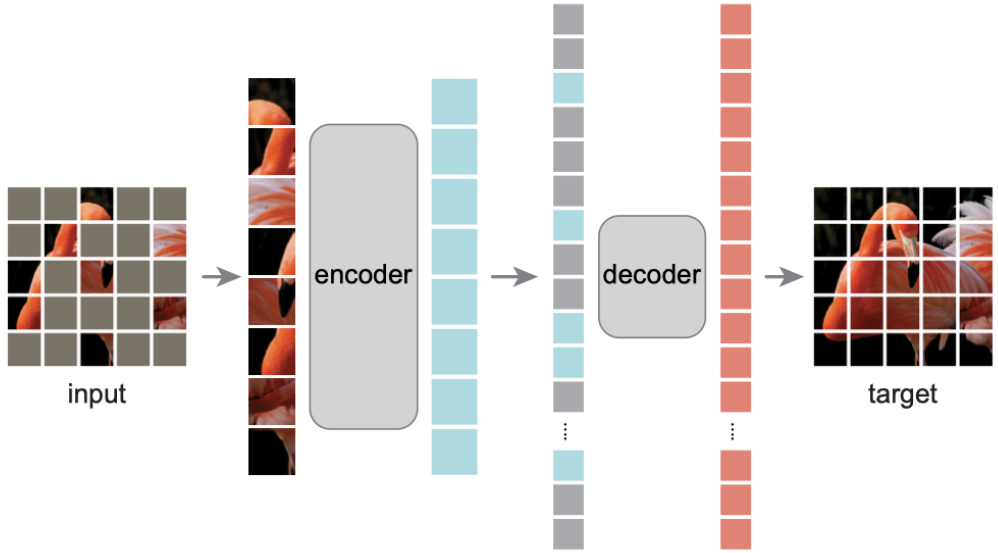
Generative Graph Self-Supervised Learning

Mar. 8, 2023

Liang Wang

Graph MAE Series

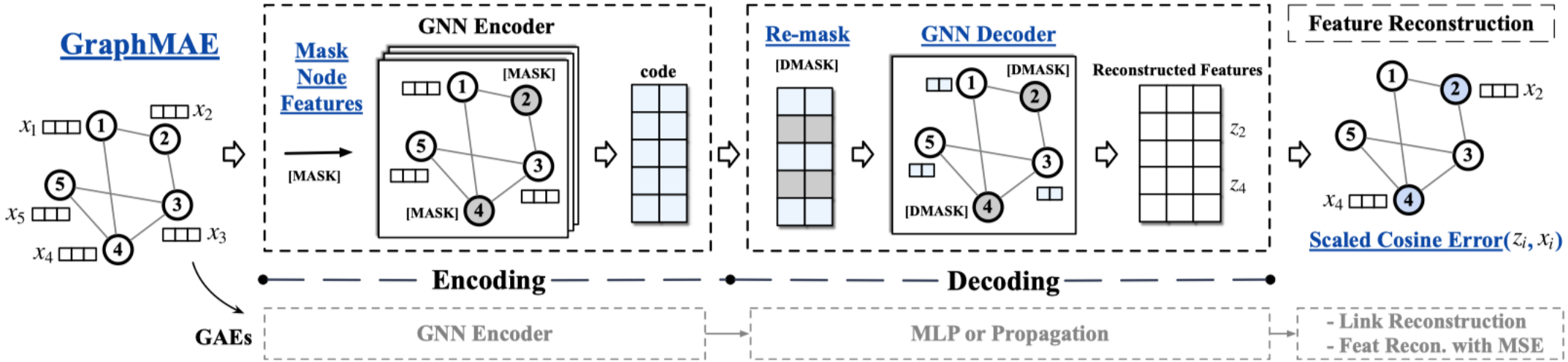
MAE, CVPR, 2022



- MAE vs Bert
  - MAE是AutoEncoder架构，而Bert不是：MAE是像素重构，需要decoder，而Bert预测标签，需要predictor
  - MAE的mask ratio更高：因为image中信息冗余更多
  - MAE的mask token不进encoder，encoder只考虑特征提取，不考虑重构（任务性质决定的，MAE中的encoder相当于特征提取器，并且微调阶段只保留encoder，要与微调阶段输入保持一致，因此encoder中不输入mask token）
- MAE vs DAE
  - 本质上属于一种去噪自编码器（DAE），但区别在于：MAE是非对称的，encoder只输入未mask部分，decoder两个部分都输入；而DAE是对称的
- 优化目标：MSE on normalized patches is better than MSE

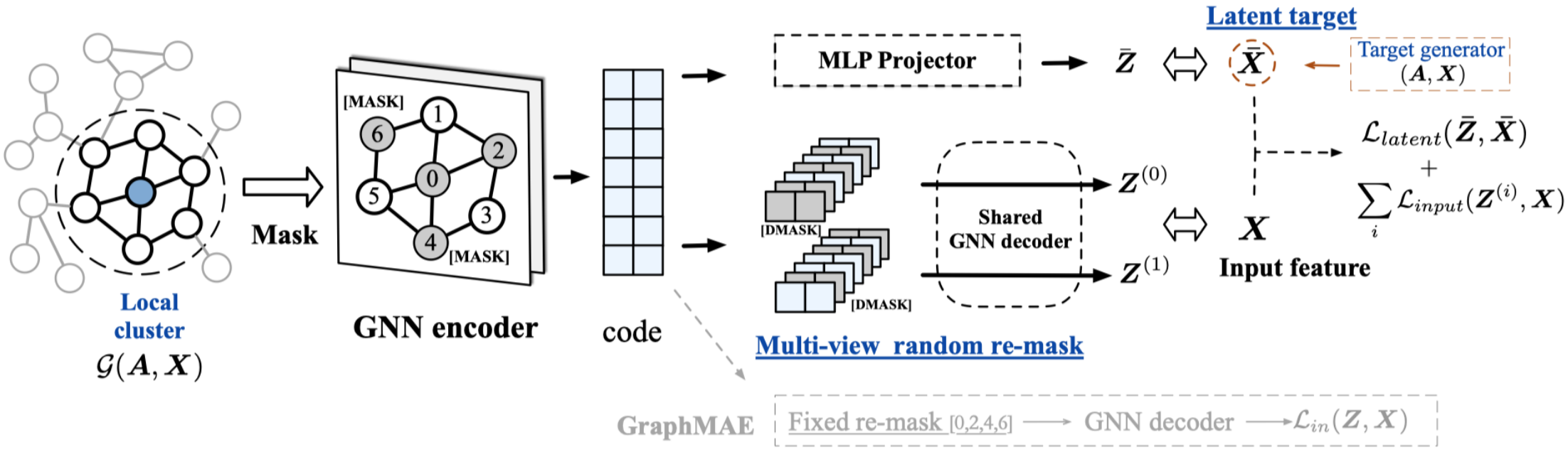
[1] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, Ross Girshick. “Masked Autoencoders Are Scalable Vision Learners.” In *CVPR*, 2022

GraphMAE, KDD, 2022



[1] "GraphMAE2: A Decoding-Enhanced Masked Self-Supervised Graph Learner." In *WWW*, 2023

GraphMAE2, WWW, 2023



[1] Zhenyu Hou, Yufei He, Yukuo Cen, Xiao Liu, Yuxiao Dong, Evgeny Kharlamov, Jie Tang. "GraphMAE2: A Decoding-Enhanced Masked Self-Supervised Graph Learner." In *WWW*. 2023

MaskGAE, arXiv, 2022

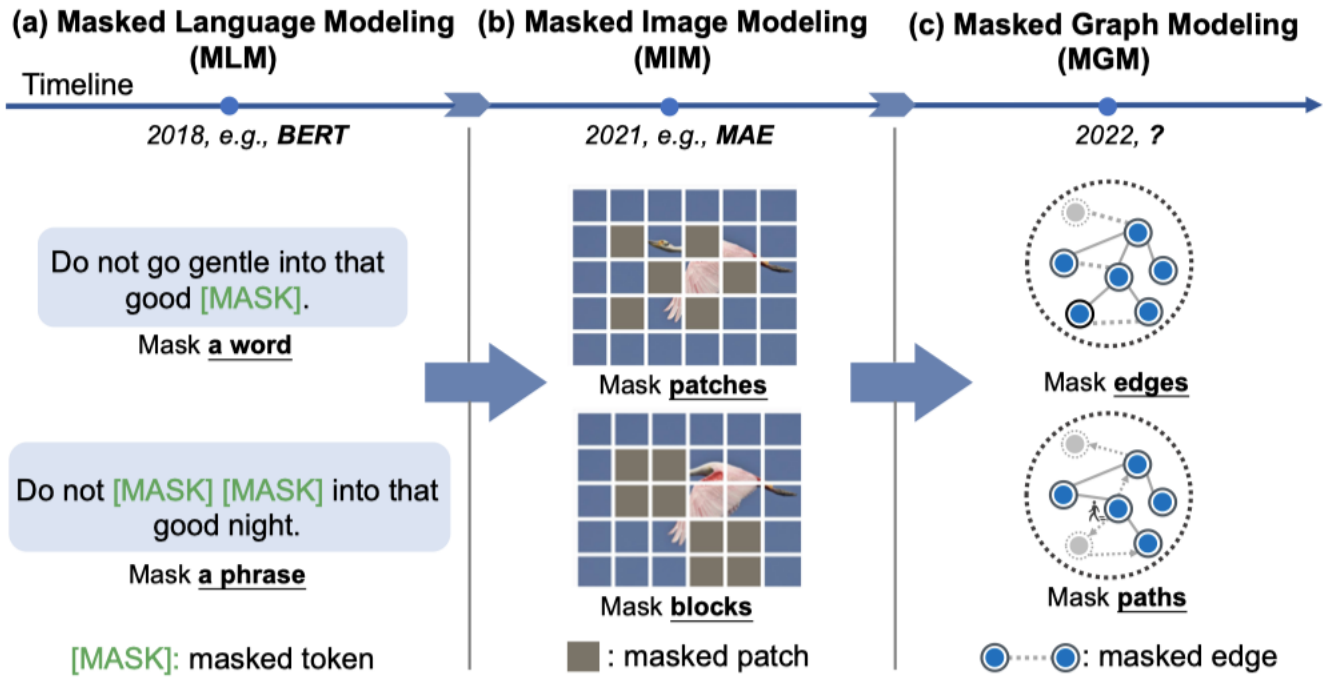


Figure 1: **From left to right**: illustrative examples of **(a)** masked language modeling (MLM), **(b)** masked image modeling (MIM), and **(c)** masked graph modeling (MGM) paradigms with different masking strategies. Similar to MLM and MIM, the goal of MGM is to learn representations by predicting randomly masked edges based on visible structure.

[1] Jintang Li, Ruofan Wu, Wangbin Sun, Liang Chen, Sheng Tian, Liang Zhu, Changhua Meng, Zibin Zheng, Weiqiang Wang. "MaskGAE: Masked Graph Modeling Meets Graph Autoencoders." arXiv. May 20, 2022

MaskGAE, arXiv, 2022

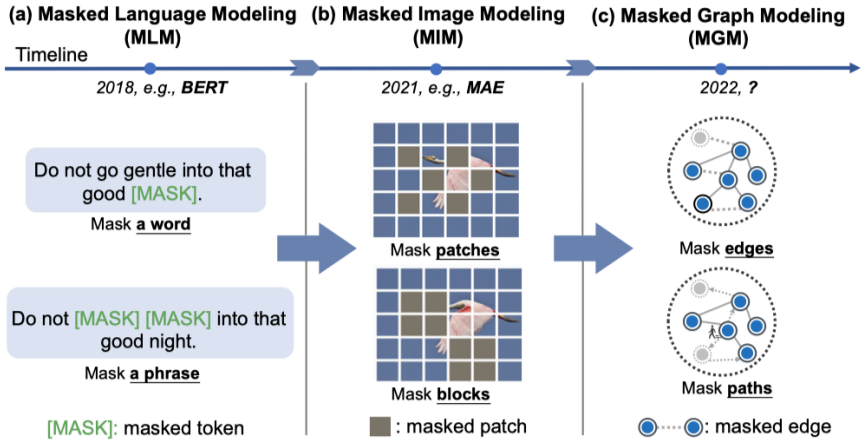


Figure 1: **From left to right**: illustrative examples of **(a)** masked language modeling (MLM), **(b)** masked image modeling (MIM), and **(c)** masked graph modeling (MGM) paradigms with different masking strategies. Similar to MLM and MIM, the goal of MGM is to learn representations by predicting randomly masked edges based on visible structure.

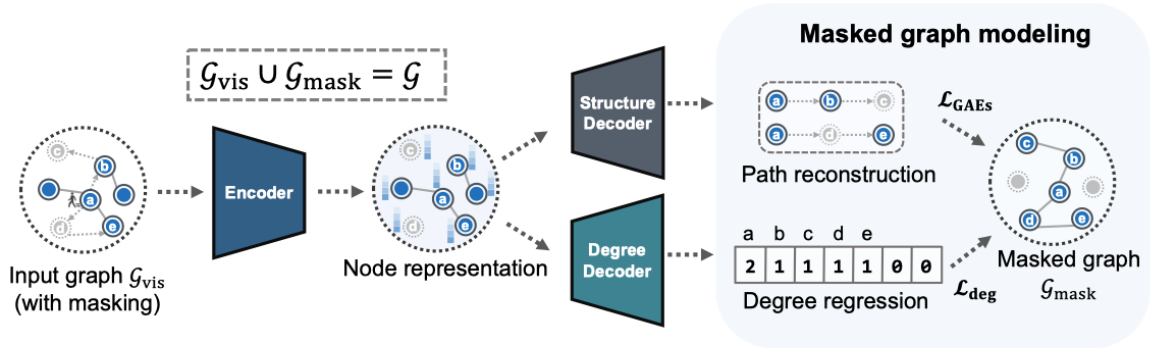


Figure 2: Overview of MaskGAE framework, performing masked graph modeling with a asymmetric encoder-decoder design. During the self-supervised learning phase, an input graph $\mathcal{G}_{\text{vis}}$ is provided, but with some of paths (a set of adjacent edges) are masked. The goal of the model is then to learn to predict the masked edges in the graph $\mathcal{G}_{\text{mask}}$ and the degree of the associated nodes, respectively.

[1] Jintang Li, Ruofan Wu, Wangbin Sun, Liang Chen, Sheng Tian, Liang Zhu, Changhua Meng, Zibin Zheng, Weiqiang Wang. “MaskGAE: Masked Graph Modeling Meets Graph Autoencoders.” arXiv. May 20, 2022

S2GAE, WSDM, 2023

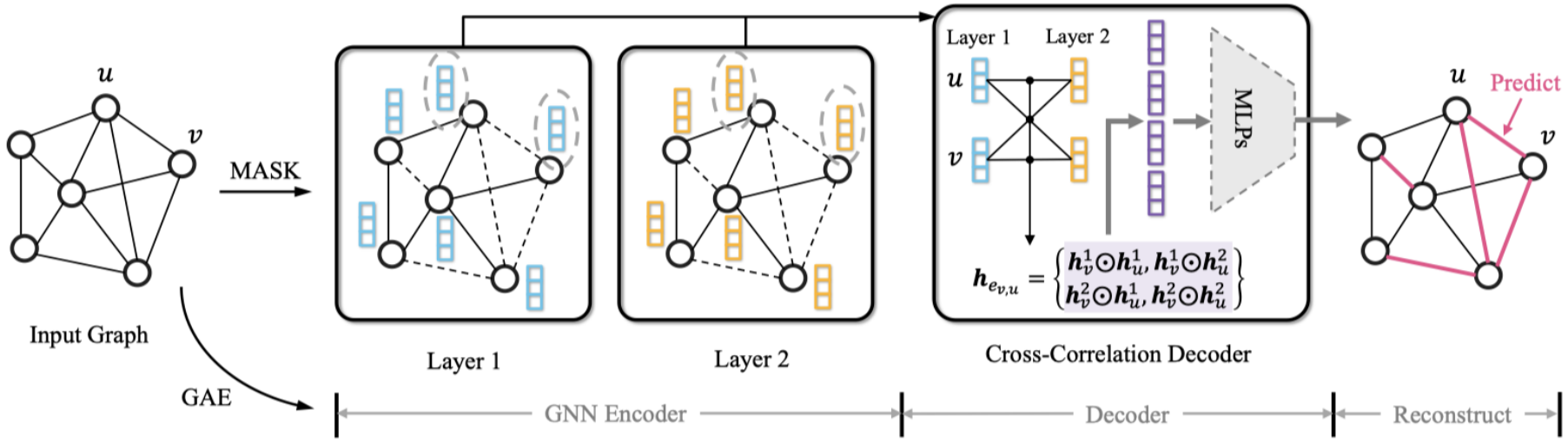


Figure 1: The proposed S2GAE architecture. Given a graph, we first apply direction-aware graph masking strategies to disturb it, obtaining perturbed graph and masked edge set. Then, the perturbed graph is fed into GNN encoder to produce hidden representations. Next, a tailored cross-correlation decoder is designed to reconstruct these masked edges by capturing the cross-correlation of their end nodes from multi-granularity representations. Finally, the whole framework is trained end-to-end by maximizing the likelihood of the masked edge set.

[1] Qiaoyu Tan, Ninghao Liu, Xiao Huang, Rui Chen, Soo-Hyun Choi, Xia Hu. "MGAE: Masked Autoencoders for Self-Supervised Learning on Graphs." arXiv. January 7, 2022

[2] Qiaoyu Tan, Ninghao Liu, Xiao Huang, Soo-Hyun Choi, Li Li, Rui Chen, Xia Hu. "S2GAE: Self-Supervised Graph Autoencoders Are Generalizable Learners with Graph Masking." In *WSDM*. 2023

GMAE, arXiv, 2022

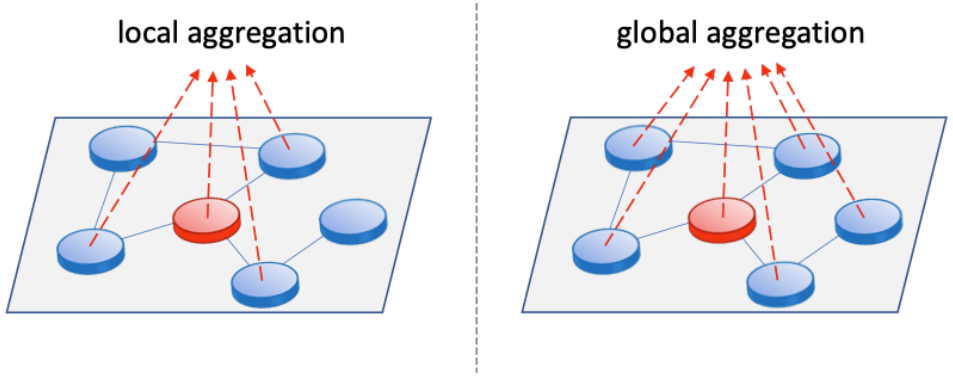


Fig. 1. An illustration of different aggregation mechanisms used by GCNs and transformers. The red node is the target node. The first picture (left) shows the local aggregation used by a GCN layer, where the embedding of the target node is obtained by aggregating its neighbors. The second picture (right) shows the global aggregation used by a transformer layer, where the embedding is obtained by aggregating all the nodes in the graph.

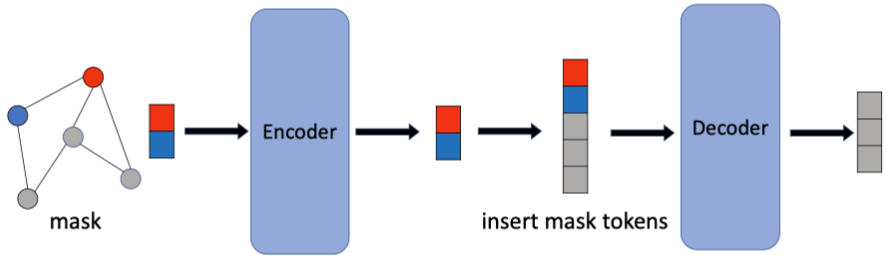


Fig. 2. GMAE framework. We first randomly mask some of the nodes (the grey nodes). The feature matrix and positional embeddings of the non-masked nodes (the red node and the blue node) are fed into an encoder to obtain the representations of the non-masked nodes. Then we insert a shared learnable mask token to represent the masked nodes. The decoder reconstructs the original features of the masked nodes.

[1] Sixiao Zhang, Hongxu Chen, Haoran Yang, Xiangguo Sun, Philip S. Yu, Guandong Xu. “Graph Masked Autoencoders with Transformers.” arXiv. May 12, 2022

HGMAE, AAAI, 2023

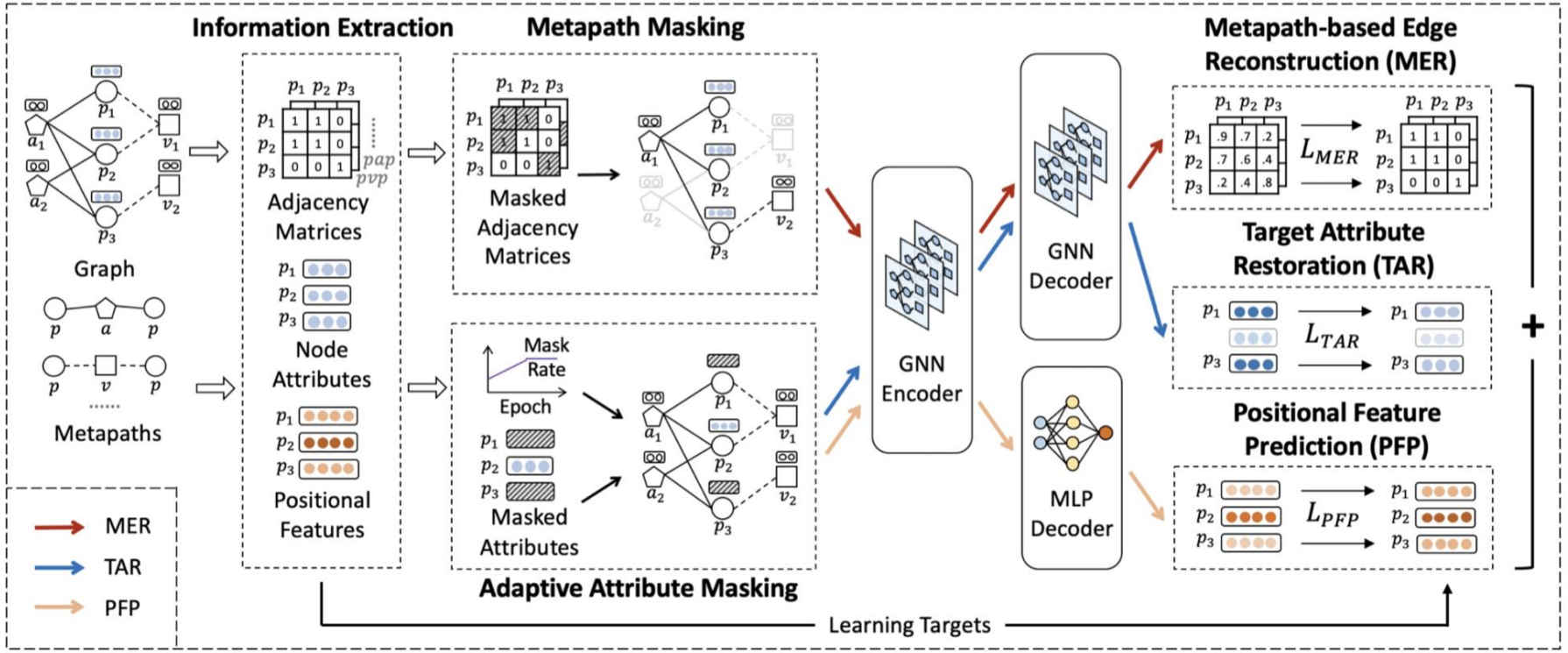


Figure 2: The overall framework of HGMAE: we first extract the metapath-based adjacency matrices, node attributes and positional features from the graph. We then design two masking techniques (i.e., metapath masking and adaptive attribute masking) to mask the inputs. Later, we feed the masked inputs into the encoder and decoders sequentially, and optimize them via three training strategies (i.e., metapath-based edge reconstruction, target attribute restoration, and positional feature prediction). The proposed strategies enable the model to capture comprehensive graph information and address identified challenges.

[1] Yijun Tian, Kaiwen Dong, Chunhui Zhang, Chuxu Zhang, Nitesh V. Chawla. "Heterogeneous Graph Masked Autoencoders." In *AAAI*. 2023

BatmanNet, arXiv, 2022

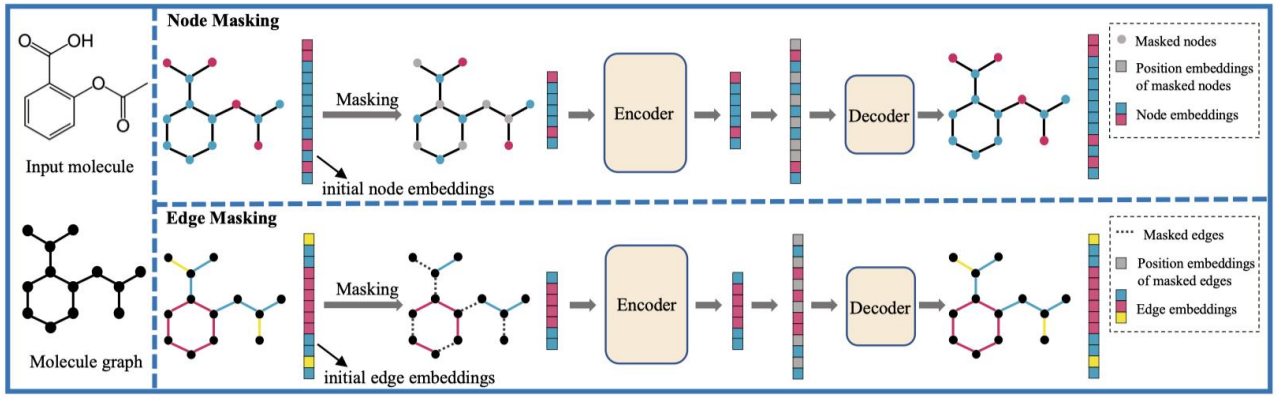
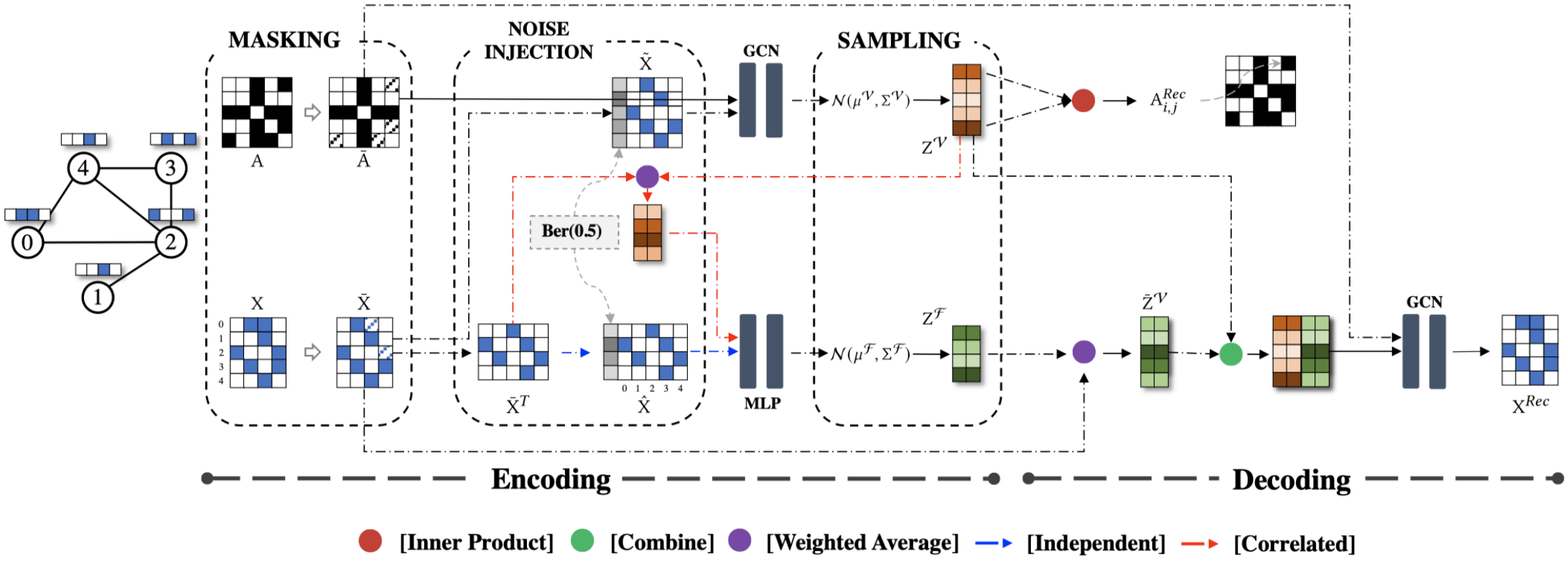


Figure 1: Illustration of the designed self-supervised tasks of BatmanNet. A very high portion of nodes or edges is randomly masked, then the BatmanNet is pre-trained to reconstruct the original molecule from the latent representation and mask tokens.

Dataset	BBBP	SIDER	ClinTox	BACE	Tox21	ToxCast	MUV	HIV
#Molecules	2039	1427	1478	1513	7831	8575	93087	41127
#tasks	1	27	2	1	12	617	17	1
TF_Robust	0.860(0.087)	0.607(0.033)	0.765(0.085)	0.824(0.022)	0.698(0.012)	0.585(0.031)	/	/
GraphConv	0.877(0.036)	0.593(0.035)	0.845(0.051)	0.854(0.011)	0.772(0.041)	0.650(0.025)	/	/
Weave	0.837(0.065)	0.543(0.034)	0.823(0.023)	0.791(0.008)	0.741(0.044)	0.678(0.024)	/	/
SchNet	0.847(0.024)	0.545(0.038)	0.717(0.042)	0.750(0.033)	0.767(0.025)	0.679(0.021)	/	/
MGCN	0.850(0.064)	0.552(0.018)	0.634(0.042)	0.734(0.030)	0.707(0.016)	0.663(0.009)	/	/
N_GRAM	0.912(0.013)	0.632(0.005)	0.855(0.037)	0.876(0.035)	0.769(0.027)	/	/	/
PretrainGNN	0.915(0.040)	0.614(0.006)	0.762(0.058)	0.851(0.027)	0.811(0.015)	0.714(0.019)	**0.839(0.012)**	0.814(0.003)
GraphMAE	0.896(0.007)	0.652(0.001)	0.850(0.007)	0.863(0.002)	0.794(0.003)	0.679(0.005)	0.796(0.008)	0.790(0.004)
$\text{GROVER}_{\text{base}}$	0.936(0.008)	0.656(0.006)	0.925(0.013)	0.878(0.016)	0.819(0.020)	0.723(0.010)	0.757(0.038)	0.764(0.029)
$\text{GROVER}_{\text{large}}$	0.940(0.019)	0.658(0.023)	**0.944(0.021)**	0.894(0.028)	0.831(0.025)	**0.737(0.010)**	0.765(0.048)	0.777(0.036)
BatmanNet	**0.944(0.007)**	**0.665(0.004)**	0.889(0.012)	**0.911(0.015)**	**0.832(0.013)**	0.733(0.009)	0.786(0.023)	**0.820(0.053)**

[1] Zhen Wang, Zheng Feng, Yanjun Li, Bowen Li, Yongrui Wang, Chulin Sha, Min He, Xiaolin Li. “BatmanNet: Bi-Branch Masked Graph Transformer Autoencoder for Molecular Representation.” arXiv. November 29, 2022

SeeGera, WWW, 2023



[1] Xiang Li, Tiandi Ye, Caihua Shan, Dongsheng Li, Ming Gao. “SeeGera: Self-Supervised Semi-Implicit Graph Variational Auto-Encoders with Masking.” In *WWW*. 2023

WGDN, AAAI, 2023

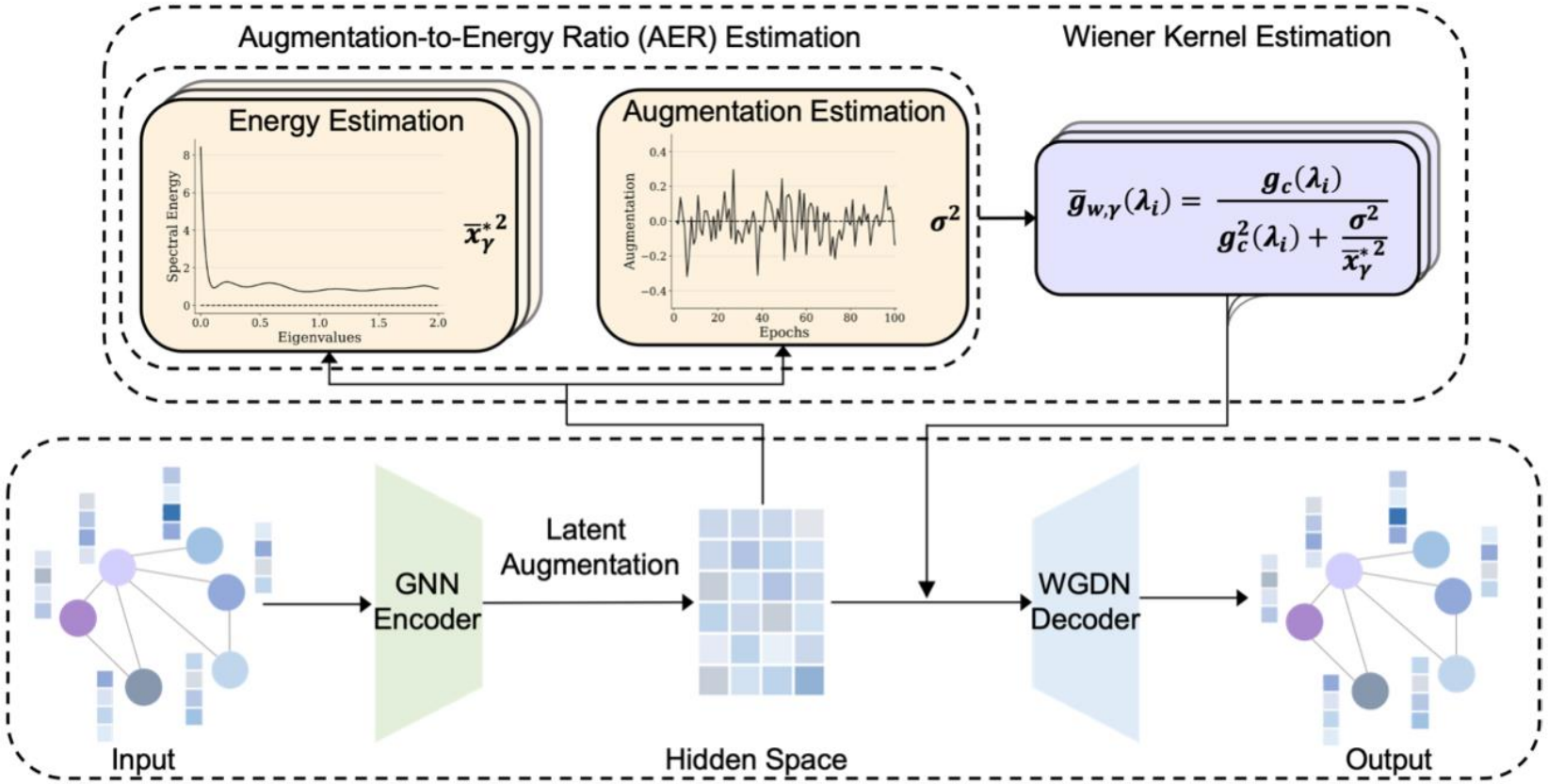


Figure 2: The autoencoder framework of WGDN for graph SSL. Given the augmented latent representations, graph wiener filter is approximated via estimating spectral energy and augmentations adaptively. With such, WGDN permits the stable feature reconstruction from the augmented latent space for representation learning.

[1] Jiashun Cheng, Man Li, Jia Li, Fugee Tsung. "Wiener Graph Deconvolutional Network Improves Graph Self-Supervised Learning." In *AAAI*. 2023

MoCE, ICLR, 2023

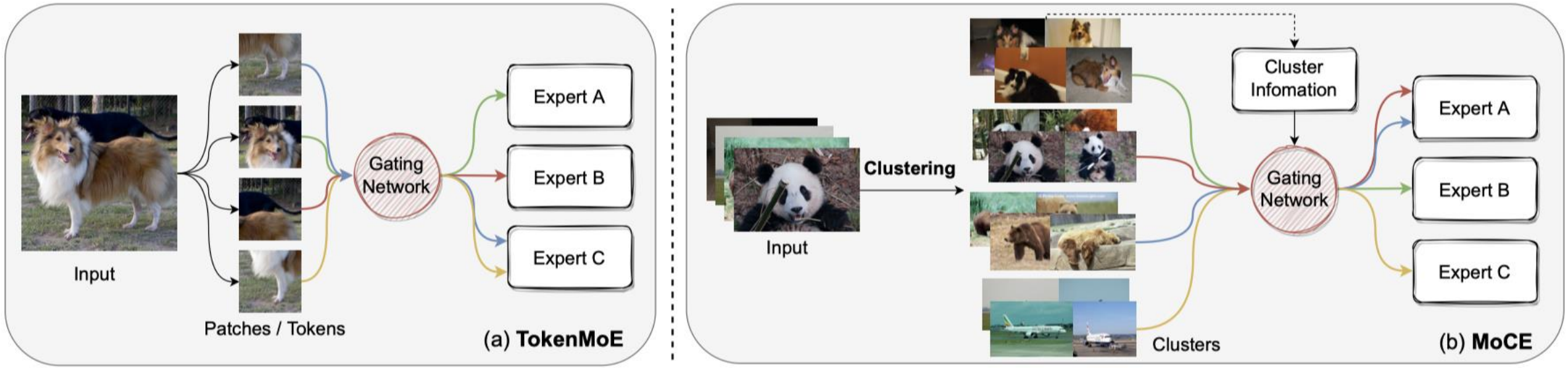
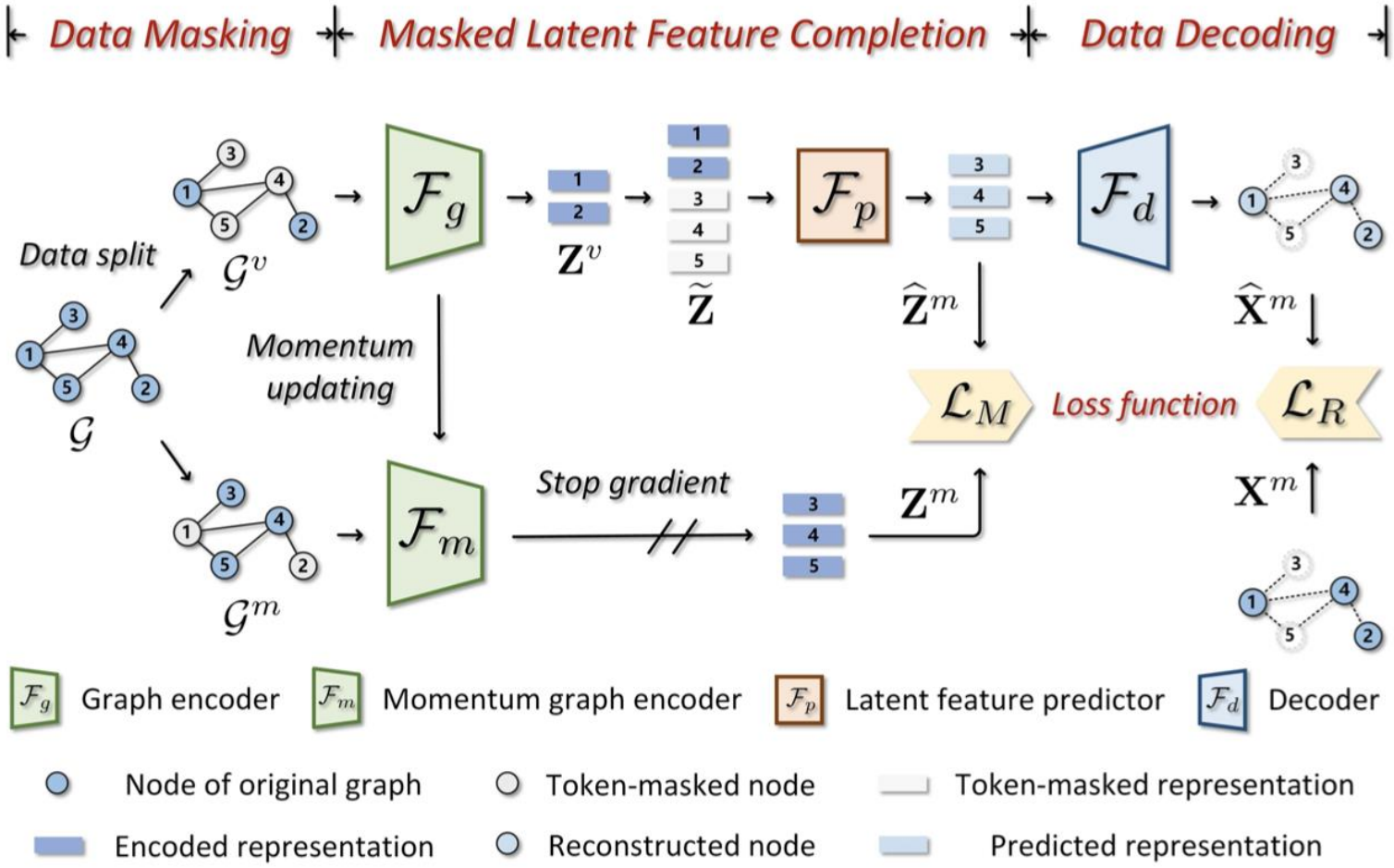


Figure 2: Model design comparison between (a) TokenMoE (Riquelme et al., 2021) and (b) MoCE. Both methods utilize the multi-expert architecture with the main difference about the input of the gating network. MoCE adopts the corresponding cluster embedding of the current token as in Eqn. 4, instead of the token embedding in Eqn. 3.2. Therefore, each expert can be trained by semantically similar images to alleviate the negative transfer phenomenon.

[1] Zhili Liu, Kai Chen, Jianhua Han, Lanqing Hong, Hang Xu, Zhenguo Li, James Kwok. "Task-Customized Masked Autoencoder via Mixture of Cluster-Conditional Experts." In *ICLR*, 2023

RARE, arXiv, 2023



[1] Wenxuan Tu, Qing Liao, Sihang Zhou, Xin Peng, Chuan Ma, Zhe Liu, Xinwang Liu, Zhiping Cai. "RARE: Robust Masked Graph Autoencoder." arXiv. April 6, 2023

Future Directions

Diffusion-based Self-supervised Learning

Theoretical Analysis

Understand MAE, arXiv, 2022

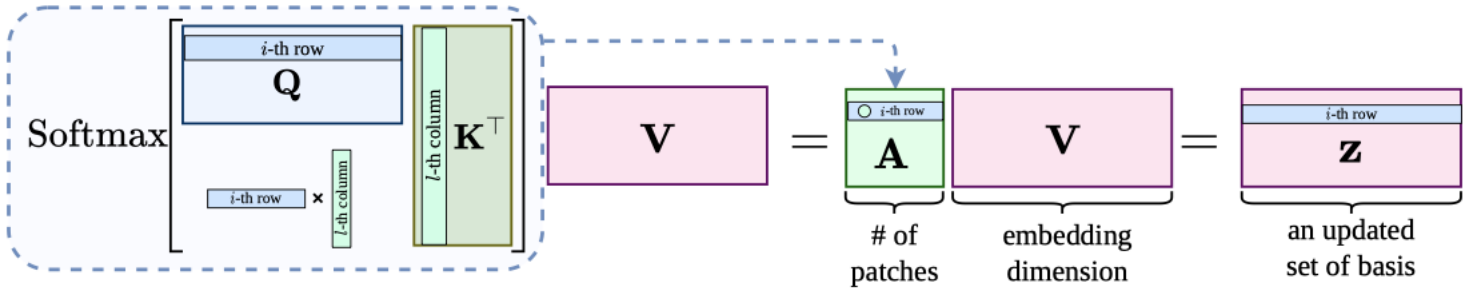
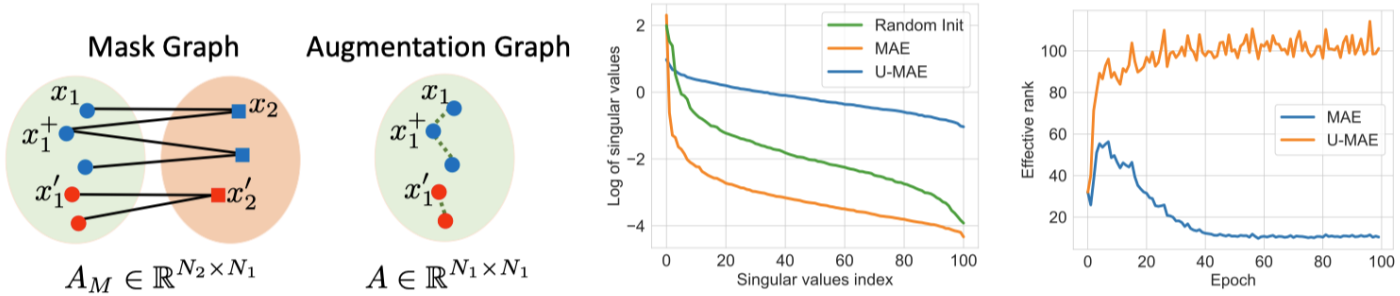


Figure 2: The i-th patch embedding from the scaled dot-product attention is a convex combination of the attention weights, which encodes the interactions among all patch embeddings. Best viewed in color.

[1] Shuhao Cao, Peng Xu, David A. Clifton. “How to Understand Masked Autoencoders.” arXiv. February 9, 2022

U-MAE, NeurIPS, 2022



(a) mask graph and augmentation graph (b) singular values (c) effective rank

Figure 1: (a) An illustration of the mask graph and the corresponding augmentation graph of MAE. Different colors denote different belonging classes. (b) Comparison of the singular values of learned features with 1) random initialization, 2) MAE loss, 3) our U-MAE loss. (c) The changing process of effective rank [24] of the encoded features trained with different objectives (MAE and U-MAE).

Theorem 3.4. *Under Assumption 3.1, MAE's reconstruction loss (Eq. (2)) can be lower bounded by the alignment loss between positive pairs* $(x_1, x_1^+) \sim \mathcal{A}(x_1, x_1^+)$,

$$\mathcal{L}_{MAE}(h) \geq \frac{1}{2}\mathcal{L}_{align}(h) - \varepsilon + const = -\frac{1}{2}\mathbb{E}_{x_1, x_1^+} h(x_1)^\top h(x_1^+) - \varepsilon + const. \tag{7}$$

$$\mathcal{L}_{\text{U-MAE}}(h) = \mathcal{L}_{\text{MAE}}(h) + \lambda \cdot \mathcal{L}_{\text{unif}}(f), \tag{10}$$

$$\text{where } \mathcal{L}_{\text{unif}}(f) = \mathbb{E}_{x_1}\mathbb{E}_{x_1^-}(f(x_1)^\top f(x_1^-))^2, \tag{11}$$

[1] Qi Zhang, Yifei Wang, Yisen Wang. "How Mask Matters: Towards Theoretical Understandings of Masked Autoencoders." In *NeurIPS*. 2022

MRP, ICLR, 2023

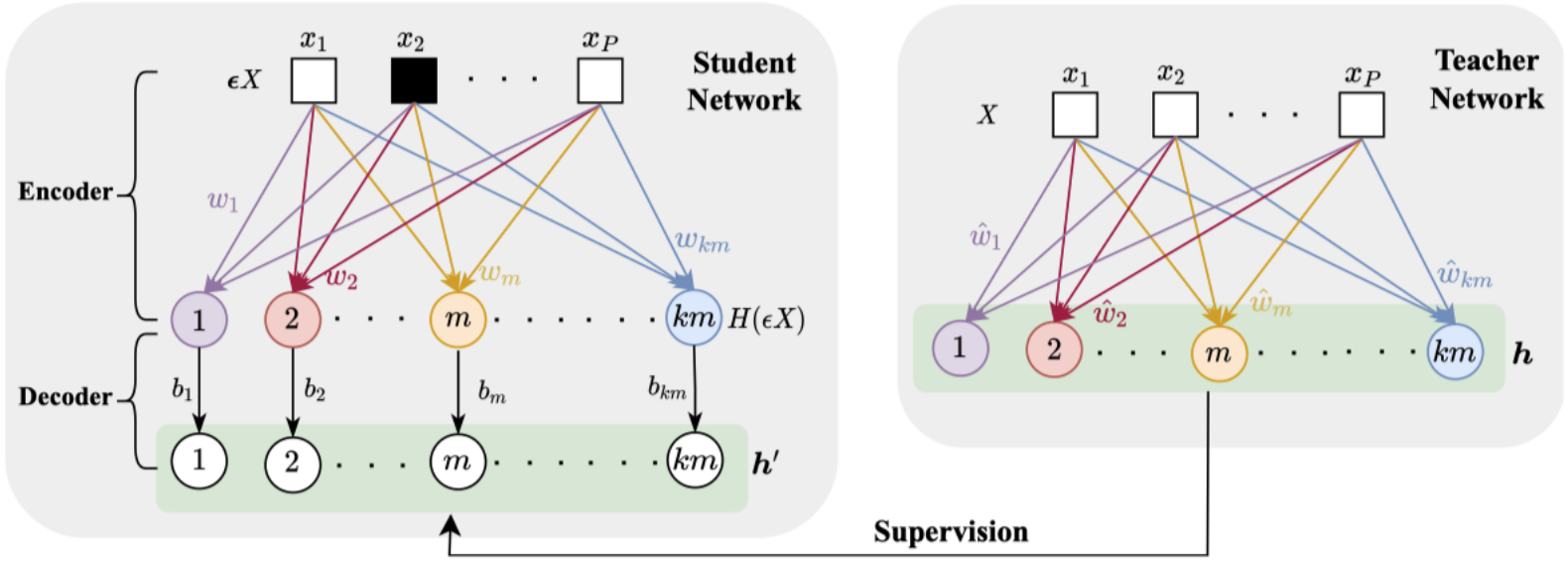


Figure 2: **Teacher-Student framework** studied in this work. Given an input $X = [x_1, \ldots, x_P]$ (image or text tokens) with P patches, this framework randomly masks patches to obtain $\boldsymbol{\epsilon} X = [\epsilon_1 x_1, \ldots, \epsilon_p x_P]$ with Bernoulli variable ϵ_p to mask, and feeds $\boldsymbol{\epsilon} X$ into student encoder H for a latent vector $H(\boldsymbol{\epsilon} X)$. Then, student decoder takes $H(\boldsymbol{\epsilon} X)$ as input and outputs $\boldsymbol{h}'$ of all patches to predict the output $\boldsymbol{h}$ of a teacher with vanilla input X as input. The encoder is two-layer CNN, and the decoder is a linear layer. For **MAE framework**, it has encoder-decoder networks and the decoder have an additional layer to map output of the encoder to recover P patches (see Fig 6 in Appendix).

[1] Jiachun Pan, Pan Zhou, Shuicheng Yan. "Towards Understanding Why Mask Reconstruction Pretraining Helps in Downstream Tasks." In *ICLR*, 2023

Test-Time Training

TTT-MAE, NeurIPS, 2022

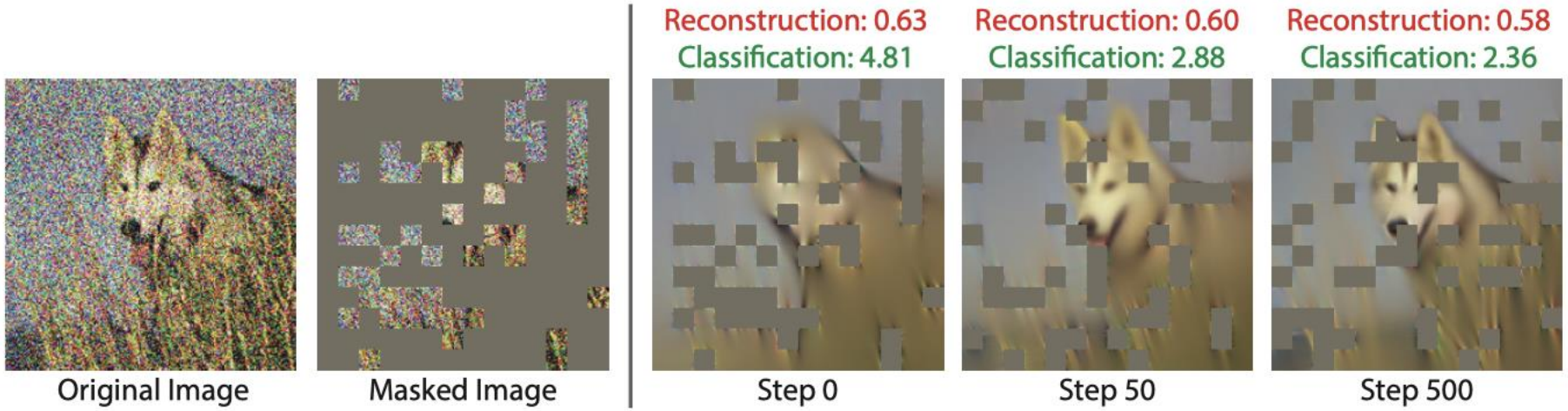


Figure 1: We train an MAE to reconstruct each test image at test time, masking 75% of the input patches. The three reconstructed images on the right visualize the progress of this one-sample learning problem. Loss of the main task (green) – object recognition – keeps dropping even after 500 steps of gradient descent, while the network continues to optimize for reconstruction (red). The unmasked patches are not shown on the right since they are not part of the reconstruction loss.

[1] Yossi Gandelsman, Yu Sun, Xinlei Chen, Alexei A. Efros. “Test-Time Training with Masked Autoencoders.” In *NeurIPS*, 2022

MAE as Data Augmentor

MRA, rejected by NeurIPS 22

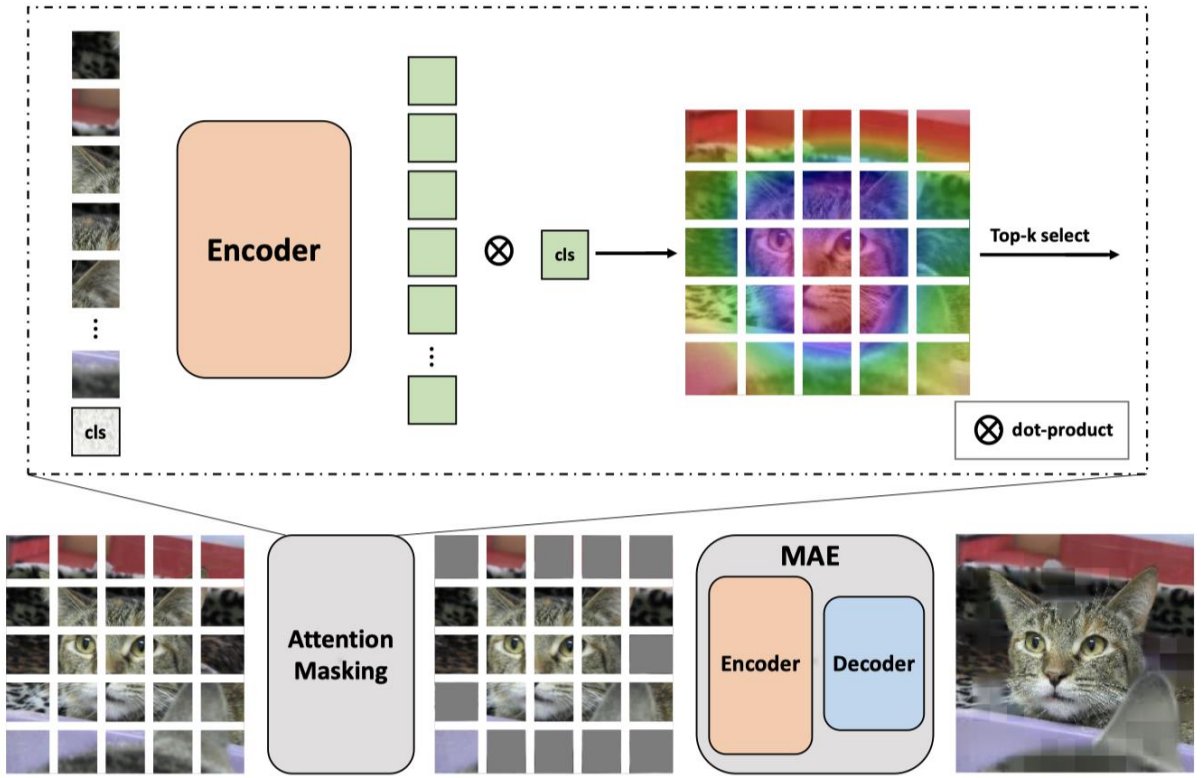


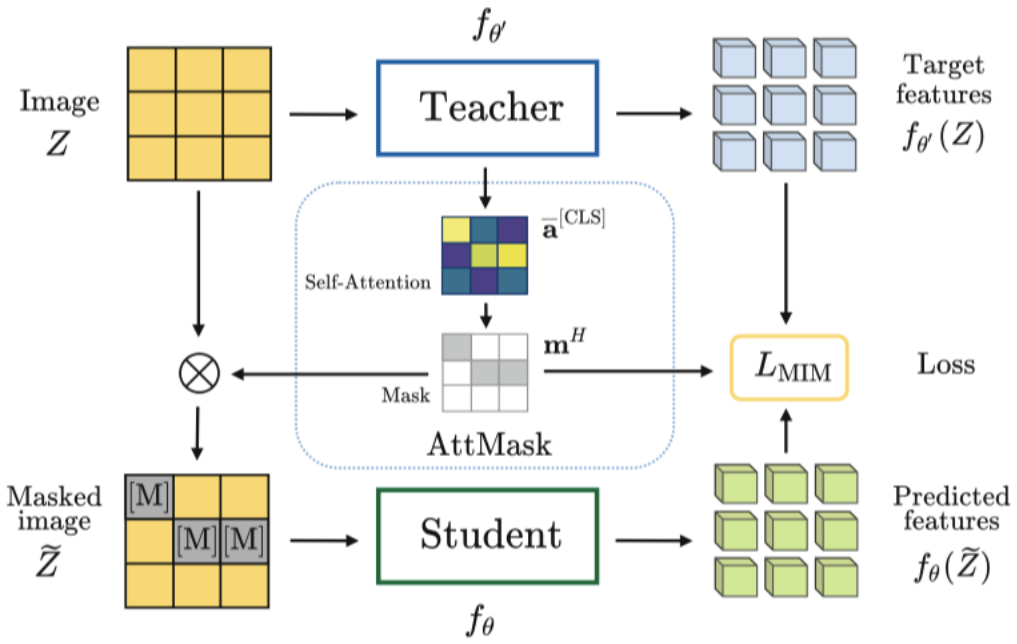
Figure 1: An overview of Mask-Reconstruct Augmentation (MRA). We first mask out the image patches via the attention-based masking strategy. Using class token as query, we calculate attention weights with the keys of each image patch. We remain patches of top-k greatest attention visible. Then, the pretrained masked autoencoder (MAE) [24] completes the original image relying on the visible image patches. The reconstructed image can be viewed as robust augmentation for a number of classification tasks, such as supervised, semi-supervised and few-shot classification.

[1] Haohang Xu, Shuangrui Ding, Xiaopeng Zhang, Hongkai Xiong, Qi Tian. “Masked Autoencoders Are Robust Data Augmentors.” arXiv. June 9, 2022

SSL Architecture

Self-distillation

AttMask, ECCV, 2022

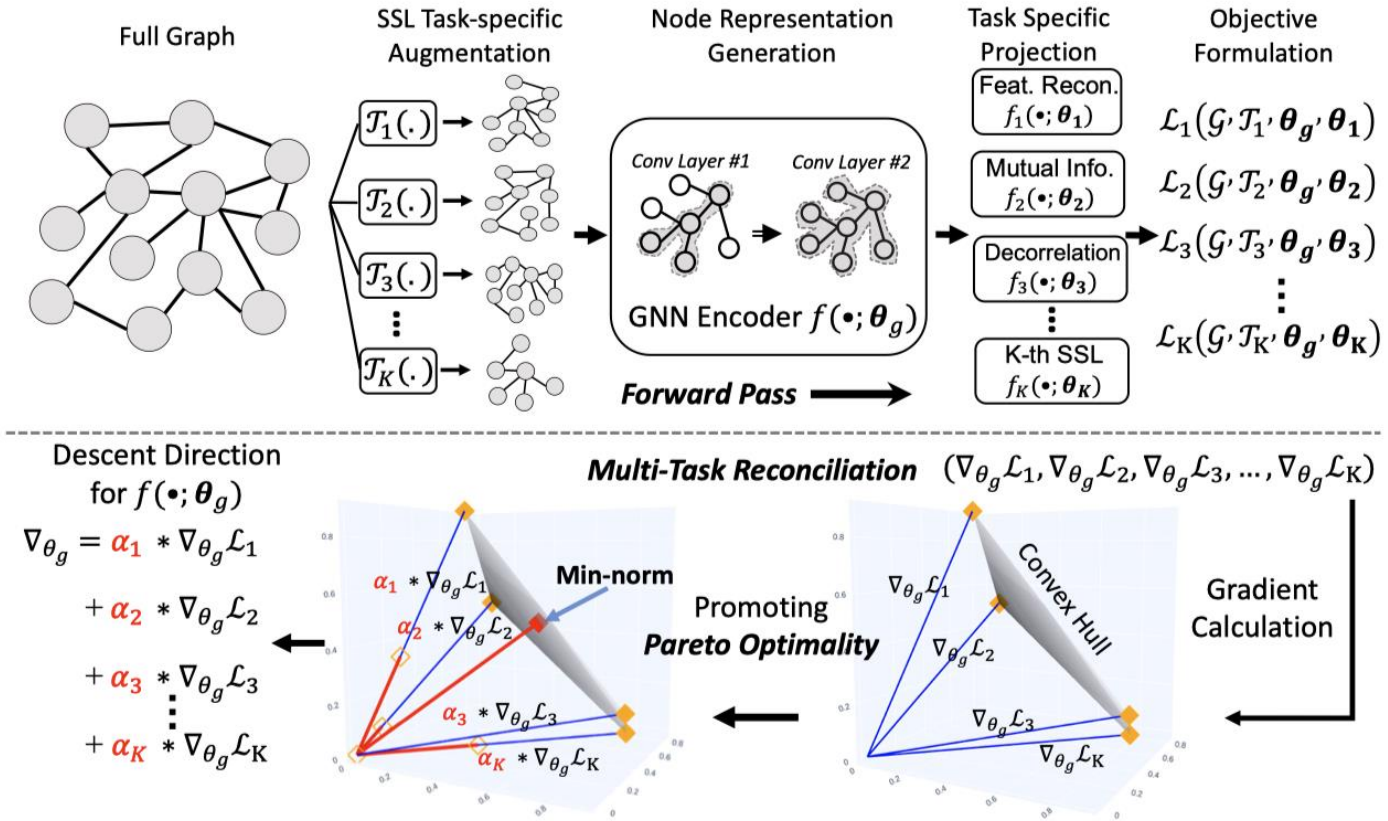


Self-distillation, following DINO

[2] Ioannis Kakogeorgiou, Spyros Gidaris, Bill Psomas, Yannis Avrithis, Andrei Bursuc, Konstantinos Karantzalos, Nikos Komodakis. “What to Hide from Your Students: Attention-Guided Masked Image Modeling.” In *ECCV*, 2022

Multi-task

ParetoGNN, ICLR, 2023



- **Motivation:**
  - **Conventional SSL methods only rely on a single philosophy, which could not generalize well on all downstream tasks.**
  - **SSL over multiple pretext tasks leads to better generalization ability and reduces the risk of overfitting.**
- **Methods:**
  - **combine multiple philosophies to enhance task generalization for SSL-based GNNs: Feature Reconstruction + Topology Reconstruction + whitening decorrelation + mutual information(node-graph) + mutual information(node-subgraph).**
  - **multiple-gradient descent algorithm to reconcile different philosophies.**

[1] Mingxuan Ju, Tong Zhao, Qianlong Wen, Wenhao Yu, Neil Shah, Yanfang Ye, Chuxu Zhang. "Multi-Task Self-Supervised Graph Neural Networks Enable Stronger Task Generalization." In *ICLR*, 2023

Multi-task

ParetoGNN, ICLR, 2023

Table 2: Performance and task generalization of PARETOGNN as well as the state-of-the-art unsupervised baselines. OOM stands for out-of-memory on a RTX3090 GPU with 24 GB memory.

Method	WIKI.CS	PUBMED	AM.PHOTO	AM.COMP.	CO.CS	CO.PHY.	CHAM.	SQUIRREL	ACTOR	RANK
AVERAGE PERFORMANCE										
DGI	72.50	59.84	83.67	77.91	85.67	79.86	61.34	50.23	33.77	4.4
GRACE	71.01	65.04	80.87	76.06	87.16	OOM	62.12	51.02	32.59	4.8
MVGRL	68.59	63.23	82.49	67.80	82.69	75.72	59.77	45.01	31.22	7.1
AUTOSSL	71.72	65.90	83.96	77.02	85.88	80.03	60.87	49.76	31.33	4.4
BGRL	74.68	67.15	80.14	79.06	87.26	81.42	60.20	49.14	32.33	3.8
CCA-SSG	73.44	64.12	83.62	78.71	83.65	79.77	62.80	51.63	35.90	3.7
GRAPHMAE	67.83	62.15	76.94	72.18	86.80	77.48	58.27	45.22	31.49	6.8
PARETOGNN	**76.03**	**72.48**	**86.58**	**82.57**	**87.80**	**83.35**	**65.21**	**55.31**	**40.76**	**1.0**
NODE CLASSIFICATION (Accuracy)										
DGI	$75.53_{\pm 0.09}$	$83.52_{\pm 0.52}$	$91.61_{\pm 0.17}$	$83.59_{\pm 0.22}$	$92.15_{\pm 0.25}$	$94.51_{\pm 0.27}$	$61.00_{\pm 1.68}$	$40.54_{\pm 0.62}$	$25.19_{\pm 0.52}$	5.9
GRACE	$80.14_{\pm 0.09}$	$86.06_{\pm 0.35}$	$92.78_{\pm 0.49}$	$89.53_{\pm 0.34}$	$91.12_{\pm 0.25}$	OOM	$58.19_{\pm 0.53}$	$41.36_{\pm 0.47}$	$24.63_{\pm 0.39}$	5.0
MVGRL	$79.11_{\pm 0.17}$	$83.62_{\pm 0.10}$	$92.48_{\pm 0.09}$	$82.78_{\pm 0.15}$	$90.33_{\pm 0.17}$	$91.05_{\pm 0.14}$	$49.87_{\pm 0.14}$	$39.81_{\pm 0.40}$	$28.01_{\pm 0.41}$	6.4
AUTOSSL	$79.55_{\pm 0.23}$	$86.26_{\pm 0.20}$	$92.71_{\pm 0.43}$	$88.76_{\pm 0.52}$	$92.17_{\pm 0.17}$	$95.13_{\pm 0.43}$	$58.94_{\pm 0.94}$	$40.63_{\pm 0.62}$	$24.54_{\pm 0.28}$	4.7
BGRL	$82.58_{\pm 0.27}$	$86.03_{\pm 0.17}$	$93.17_{\pm 0.23}$	$90.15_{\pm 0.18}$	$91.77_{\pm 0.47}$	$95.73_{\pm 0.23}$	$56.05_{\pm 1.06}$	$41.64_{\pm 0.79}$	$24.03_{\pm 0.78}$	4.1
CCA-SSG	$82.48_{\pm 0.35}$	$86.36_{\pm 0.35}$	$93.29_{\pm 0.49}$	$89.58_{\pm 0.70}$	**$94.24_{\pm 0.17}$**	$95.63_{\pm 0.09}$	$57.39_{\pm 1.38}$	$42.22_{\pm 0.94}$	$26.35_{\pm 0.35}$	2.8
GRAPHMAE	$77.12_{\pm 0.30}$	$83.91_{\pm 0.26}$	$90.71_{\pm 0.40}$	$79.44_{\pm 0.48}$	$93.13_{\pm 0.15}$	**$95.79_{\pm 0.06}$**	$55.50_{\pm 0.82}$	$35.87_{\pm 0.61}$	**$28.97_{\pm 0.27}$**	5.3
PARETOGNN	**$82.87_{\pm 0.13}$**	**$87.03_{\pm 0.25}$**	**$93.85_{\pm 0.28}$**	**$90.75_{\pm 0.17}$**	$92.21_{\pm 0.14}$	$95.45_{\pm 0.10}$	**$63.13_{\pm 0.84}$**	**$46.60_{\pm 1.08}$**	$26.62_{\pm 0.67}$	**1.8**
NODE CLUSTERING (NMI)										
DGI	$44.35_{\pm 0.12}$	$9.68_{\pm 0.31}$	$60.31_{\pm 0.23}$	$47.76_{\pm 0.02}$	$72.88_{\pm 0.21}$	$58.76_{\pm 0.43}$	$6.99_{\pm 1.74}$	$2.16_{\pm 0.14}$	$1.49_{\pm 0.05}$	5.1
GRACE	$40.40_{\pm 0.10}$	$25.64_{\pm 0.24}$	$55.20_{\pm 0.70}$	$41.77_{\pm 0.32}$	**$76.61_{\pm 0.26}$**	OOM	$11.73_{\pm 1.01}$	$2.53_{\pm 0.31}$	$1.09_{\pm 0.22}$	5.3
MVGRL	$36.20_{\pm 0.29}$	$24.87_{\pm 0.13}$	$56.36_{\pm 0.08}$	$31.27_{\pm 0.29}$	$73.34_{\pm 0.01}$	$58.27_{\pm 0.01}$	**$18.57_{\pm 0.26}$**	**$4.40_{\pm 0.21}$**	**$2.63_{\pm 0.06}$**	4.3
AUTOSSL	$36.99_{\pm 0.21}$	$28.99_{\pm 0.26}$	$64.06_{\pm 0.65}$	$41.85_{\pm 0.36}$	$74.04_{\pm 0.22}$	$55.23_{\pm 0.18}$	$9.67_{\pm 1.21}$	$2.11_{\pm 0.12}$	$1.35_{\pm 0.11}$	5.0
BGRL	$44.95_{\pm 0.32}$	$26.38_{\pm 0.36}$	$50.56_{\pm 0.24}$	$44.04_{\pm 0.36}$	$74.06_{\pm 0.51}$	**$61.04_{\pm 0.11}$**	$11.29_{\pm 1.45}$	$2.28_{\pm 0.53}$	$1.32_{\pm 0.07}$	4.3
CCA-SSG	$44.17_{\pm 0.04}$	$27.15_{\pm 1.56}$	$64.06_{\pm 0.03}$	$49.78_{\pm 0.01}$	$67.14_{\pm 0.49}$	$54.33_{\pm 2.69}$	$12.17_{\pm 0.79}$	$3.02_{\pm 0.04}$	$1.01_{\pm 0.02}$	4.3
GRAPHMAE	$35.73_{\pm 0.04}$	$19.00_{\pm 0.11}$	$51.42_{\pm 0.46}$	$43.51_{\pm 0.16}$	$76.18_{\pm 0.59}$	$46.90_{\pm 0.07}$	$8.66_{\pm 0.47}$	$3.70_{\pm 0.32}$	$1.24_{\pm 0.05}$	5.7
PARETOGNN	**$47.52_{\pm 0.29}$**	**$34.74_{\pm 0.06}$**	**$68.25_{\pm 1.25}$**	**$52.53_{\pm 0.34}$**	$74.94_{\pm 0.98}$	$60.43_{\pm 0.13}$	$14.49_{\pm 0.23}$	$2.81_{\pm 0.23}$	$1.53_{\pm 0.04}$	**1.9**
LINK PREDICTION (AUC)										
DGI	$94.36_{\pm 0.04}$	$78.59_{\pm 0.58}$	$94.24_{\pm 0.16}$	$90.37_{\pm 0.03}$	$93.46_{\pm 0.12}$	$88.82_{\pm 0.05}$	$91.64_{\pm 0.62}$	$92.45_{\pm 0.01}$	$72.01_{\pm 0.41}$	5.0
GRACE	$92.32_{\pm 0.05}$	$87.44_{\pm 1.03}$	$91.82_{\pm 0.12}$	$88.58_{\pm 0.08}$	$97.00_{\pm 0.02}$	OOM	$93.81_{\pm 0.47}$	$93.57_{\pm 0.06}$	$82.31_{\pm 0.17}$	4.6
MVGRL	$94.34_{\pm 0.20}$	$90.31_{\pm 0.02}$	$93.86_{\pm 0.25}$	$75.15_{\pm 0.34}$	$92.44_{\pm 0.04}$	$87.54_{\pm 0.08}$	$94.50_{\pm 0.44}$	$88.81_{\pm 0.33}$	$77.93_{\pm 0.07}$	5.0
AUTOSSL	$93.86_{\pm 0.02}$	$86.84_{\pm 1.30}$	$95.57_{\pm 0.13}$	$93.99_{\pm 0.03}$	$95.71_{\pm 0.15}$	$95.93_{\pm 0.07}$	$90.05_{\pm 0.78}$	$93.84_{\pm 0.19}$	$70.51_{\pm 0.29}$	4.3
BGRL	$94.31_{\pm 0.06}$	$94.35_{\pm 0.02}$	$93.33_{\pm 0.06}$	$93.59_{\pm 0.05}$	$97.37_{\pm 0.04}$	$93.38_{\pm 0.07}$	$92.24_{\pm 0.62}$	$83.60_{\pm 0.10}$	$76.91_{\pm 0.10}$	4.4
CCA-SSG	$90.92_{\pm 0.33}$	$73.53_{\pm 1.50}$	$89.47_{\pm 0.13}$	$86.72_{\pm 0.13}$	$91.77_{\pm 0.03}$	$93.64_{\pm 0.02}$	$95.43_{\pm 0.31}$	$87.30_{\pm 0.25}$	$78.14_{\pm 0.09}$	5.7
GRAPHMAE	$89.47_{\pm 0.02}$	$86.24_{\pm 0.19}$	$83.33_{\pm 0.07}$	$84.65_{\pm 1.08}$	$96.48_{\pm 0.17}$	$96.57_{\pm 0.08}$	$89.40_{\pm 1.48}$	$86.48_{\pm 0.14}$	$80.65_{\pm 0.28}$	5.9
PARETOGNN	**$96.48_{\pm 0.01}$**	**$94.58_{\pm 0.02}$**	**$96.08_{\pm 0.08}$**	**$97.16_{\pm 0.04}$**	**$98.18_{\pm 0.01}$**	**$98.33_{\pm 0.12}$**	**$95.78_{\pm 0.05}$**	**$96.46_{\pm 0.05}$**	**$84.29_{\pm 0.04}$**	**1.0**
PARTITION PREDICTION (Accuracy)										
DGI	$75.75_{\pm 0.19}$	$67.58_{\pm 0.01}$	**$88.50_{\pm 0.20}$**	**$89.91_{\pm 0.07}$**	$84.19_{\pm 0.66}$	$77.34_{\pm 0.01}$	$85.75_{\pm 0.14}$	$65.77_{\pm 0.02}$	$36.40_{\pm 0.02}$	2.9
GRACE	$71.19_{\pm 0.13}$	$61.01_{\pm 1.57}$	$83.66_{\pm 0.83}$	$84.36_{\pm 0.21}$	$83.89_{\pm 0.82}$	OOM	$84.76_{\pm 2.03}$	$66.64_{\pm 0.12}$	$22.32_{\pm 0.05}$	5.4
MVGRL	$64.73_{\pm 0.25}$	$54.11_{\pm 0.06}$	$87.25_{\pm 0.44}$	$81.99_{\pm 0.04}$	$74.66_{\pm 0.19}$	$66.03_{\pm 0.35}$	$76.12_{\pm 1.03}$	$47.03_{\pm 0.81}$	$16.31_{\pm 0.04}$	7.2
AUTOSSL	$76.47_{\pm 0.07}$	$61.52_{\pm 0.01}$	$83.51_{\pm 0.05}$	$83.46_{\pm 0.79}$	$81.59_{\pm 0.80}$	$73.85_{\pm 0.04}$	$84.83_{\pm 0.94}$	$62.45_{\pm 0.02}$	$28.90_{\pm 0.04}$	5.0
BGRL	$76.87_{\pm 0.03}$	$61.84_{\pm 0.02}$	$83.51_{\pm 0.05}$	$88.45_{\pm 0.38}$	$85.84_{\pm 1.29}$	$75.53_{\pm 0.01}$	$81.21_{\pm 0.99}$	$69.04_{\pm 0.39}$	$27.07_{\pm 0.04}$	3.9
CCA-SSG	$76.18_{\pm 0.46}$	$69.46_{\pm 0.26}$	$87.67_{\pm 0.31}$	$88.77_{\pm 0.19}$	$81.44_{\pm 0.23}$	$75.49_{\pm 0.15}$	$86.22_{\pm 0.66}$	$73.97_{\pm 0.22}$	$38.11_{\pm 0.29}$	3.1
GRAPHMAE	$69.01_{\pm 0.27}$	$59.46_{\pm 0.08}$	$82.31_{\pm 0.13}$	$81.11_{\pm 1.57}$	$81.39_{\pm 0.20}$	$70.67_{\pm 0.16}$	$79.50_{\pm 1.26}$	$54.82_{\pm 0.67}$	$15.09_{\pm 0.01}$	7.2
PARETOGNN	**$77.23_{\pm 0.36}$**	**$73.57_{\pm 0.23}$**	$88.13_{\pm 0.39}$	$89.84_{\pm 0.06}$	**$85.89_{\pm 0.33}$**	**$79.19_{\pm 0.20}$**	**$87.43_{\pm 1.05}$**	**$75.39_{\pm 0.65}$**	**$50.61_{\pm 0.75}$**	**1.2**

[1] Mingxuan Ju, Tong Zhao, Qianlong Wen, Wenhao Yu, Neil Shah, Yanfang Ye, Chuxu Zhang. "Multi-Task Self-Supervised Graph Neural Networks Enable Stronger Task Generalization." In *ICLR*, 2023

Multi-task

AutoSSL, ICLR, 2022

AUTOMATED SELF-SUPERVISED LEARNING FOR GRAPHS

Wei Jin *
Michigan State University
jinwei2@msu.edu

Xiaorui Liu
Michigan State University
xiaorui@msu.edu

Xiaoyu Zhao
City University of Hong Kong
xy.zhao@cityu.edu.hk

Yao Ma
New Jersey Institute of Technology
yao.ma@njit.edu

Neil Shah
Snap Inc.
nshah@snap.com

Jiliang Tang
Michigan State University
tangjili@msu.edu

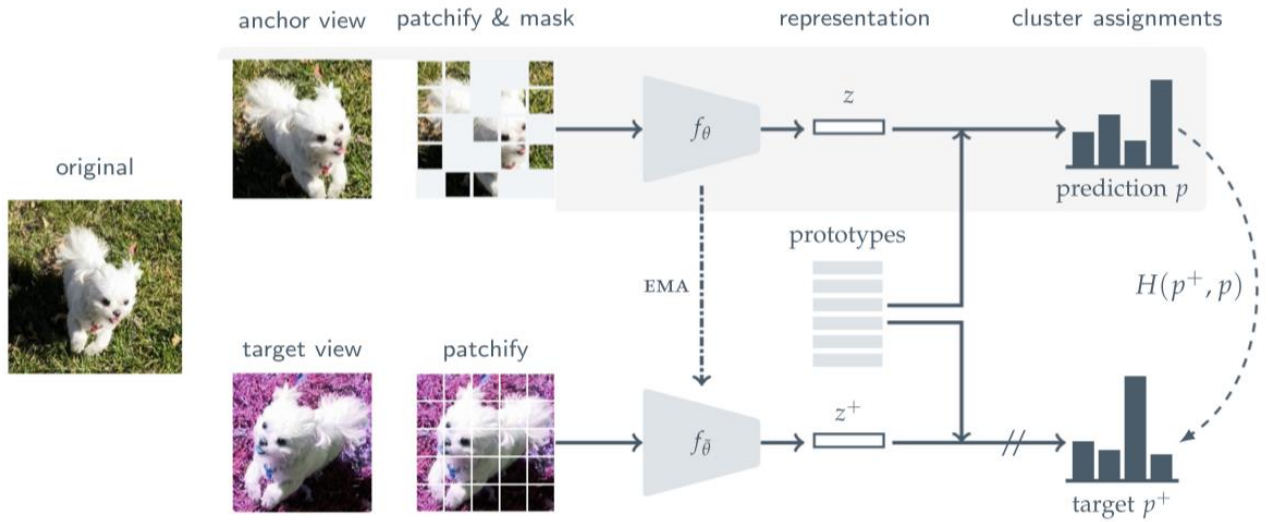
ABSTRACT

Graph self-supervised learning has gained increasing attention due to its capacity to learn expressive node representations. Many pretext tasks, or loss functions have been designed from distinct perspectives. However, we observe that different pretext tasks affect downstream tasks differently across datasets, which suggests that searching over pretext tasks is crucial for graph self-supervised learning. Different from existing works focusing on designing single pretext tasks, this work aims to investigate how to automatically leverage multiple pretext tasks effectively. Nevertheless, evaluating representations derived from multiple pretext tasks without direct access to ground truth labels makes this problem challenging. To address this obstacle, we make use of a key principle of many real-world graphs, i.e., homophily, or the principle that "like attracts like," as the guidance to effectively search various self-supervised pretext tasks. We provide theoretical understanding and empirical evidence to justify the flexibility of homophily in this search task. Then we propose the AUTOSSL framework to automatically search over combinations of various self-supervised tasks. By evaluating the framework on 8 real-world datasets, our experimental results show that AUTOSSL can significantly boost the performance on downstream tasks including node clustering and node classification compared with training under individual tasks.

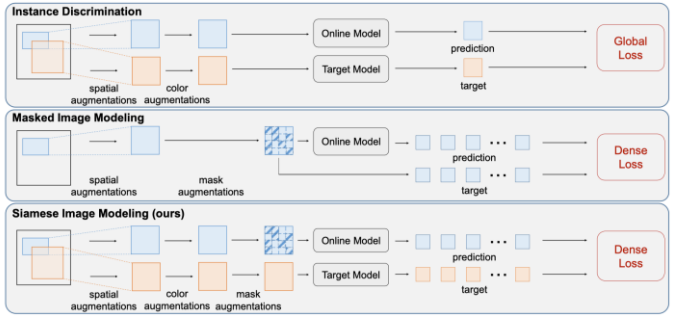
[1] Wei Jin, Xiaorui Liu, Xiaoyu Zhao, Yao Ma, Neil Shah, Jiliang Tang. "Automated Self-Supervised Learning for Graphs." In *ICLR*, 2022

Masked Siamese Networks for Label-Efficient Fine-Tuning

MSN, ECCV, 2022



Related work (Siamese Image Modeling):
[1] Chenxin Tao, Xizhou Zhu, Weijie Su, Gao Huang, Bin Li, Jie Zhou, Yu Qiao, Xiaogang Wang, Jifeng Dai. “Siamese Image Modeling for Self-Supervised Vision Representation Learning.” arXiv. 2022



[1] Ioannis Kakogeorgiou, Spyros Gidaris, Bill Psomas, Yannis Avrithis, Andrei Bursuc, Konstantinos Karantzalos, Nikos Komodakis. “What to Hide from Your Students: Attention-Guided Masked Image Modeling.” In *ECCV*, 2022

Maximizing Rate Reduction

G^2R, WWW, 2023

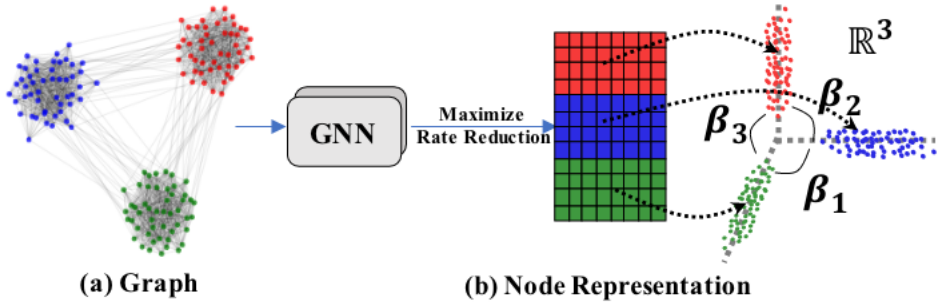


Figure 1: Overview of G^2R. It maps nodes in distinct groups (implicitly preserved in adjacency matrix) into different subspaces, while each subspace is compact and different subspaces are dispersedly distributed. Different colors indicate different subspaces and node groups.

[1] Xiaotian Han, Zhimeng Jiang, Ninghao Liu, Qingquan Song, Jundong Li, Xia Hu. “Geometric Graph Representation Learning via Maximizing Rate Reduction.” In *WWW*. 2022

Masking Strategy

Masking mechanisms define the specific pretext task, i.e., what kind of information is to be exploited and what kind of information is to be predicted.

What to Mask?

- Random Masking
- Adversarial-based Masking
- Attention-based Masking
- Semantic-based Masking

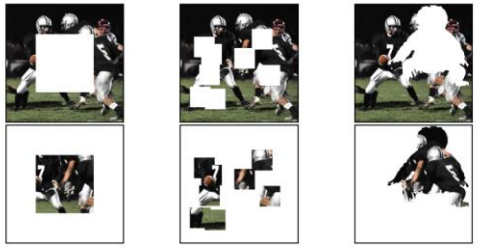
How to Mask?

- Zeros-replacement, Noise-replacement, other-feature-replacement, learned shared token
- Corruption-based
- Mix-based

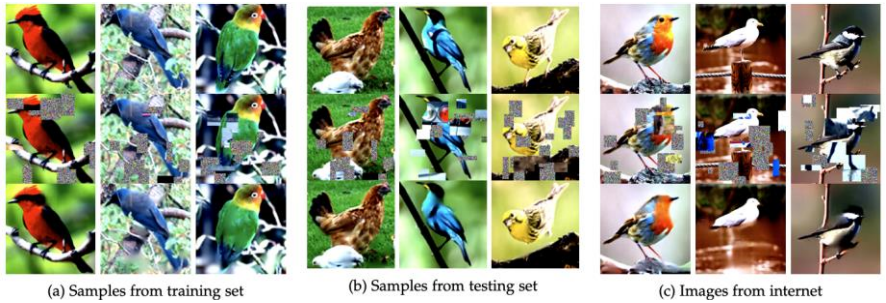
What to Mask?

- Random Masking
- Adversarial-based Masking
- Attention-based Masking
- Semantic-based Masking

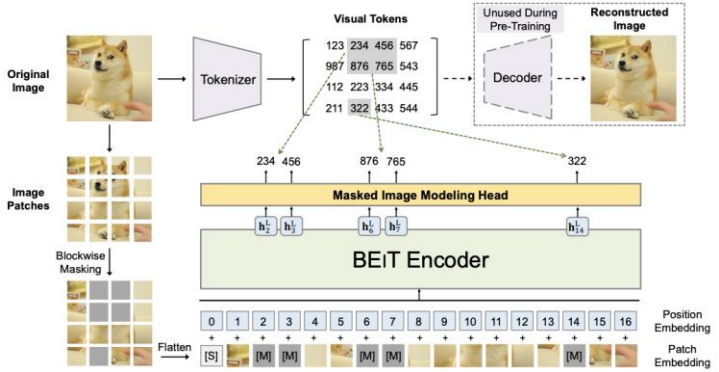
Masking Strategies in MIM - Random



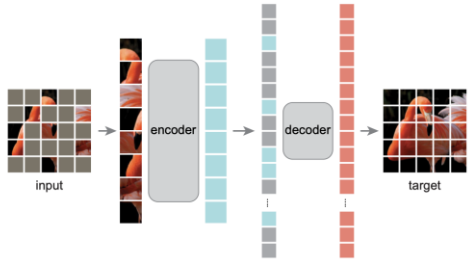
Context Encoder, CVPR16
Central region mask, random shape mask



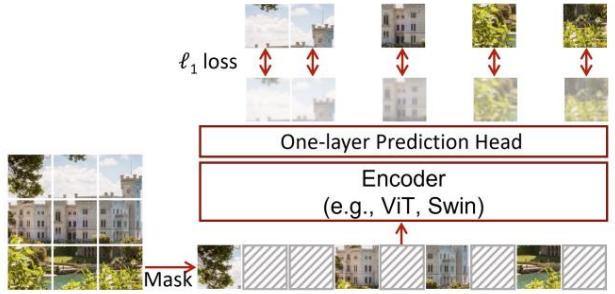
SiT, arXiv21
Random block mask



BEiT, ICLR22
Random block mask



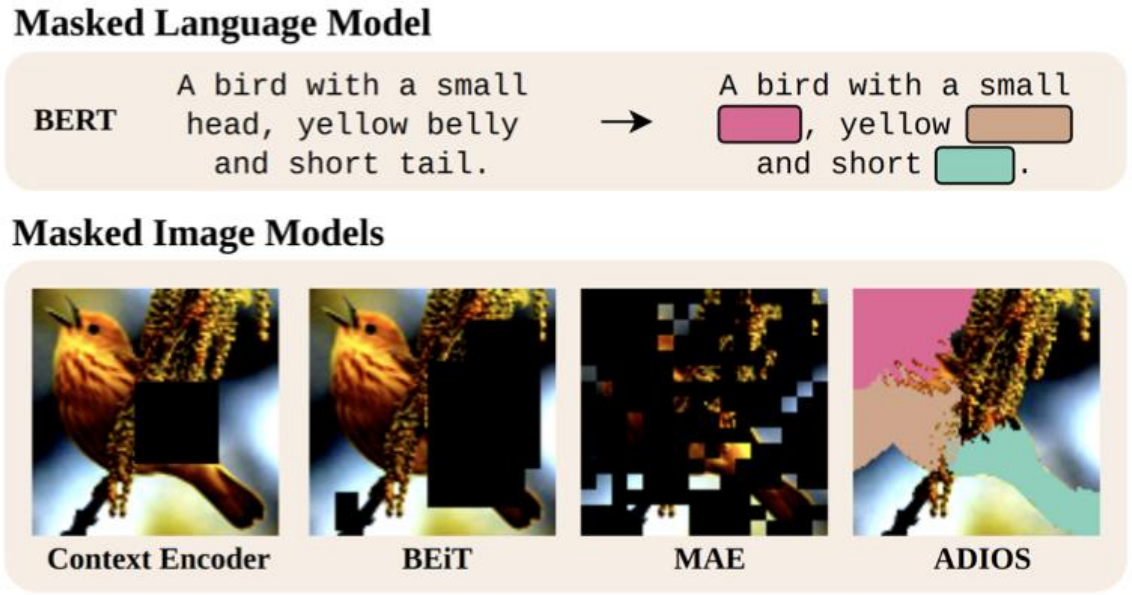
MAE, CVPR22
Random patch mask



SimMIM, CVPR22
Random patch mask

[1] Deepak Pathak, Philipp Krähenbühl, Jeff Donahue, Trevor Darrell, Alexei A. Efros. "Context Encoders: Feature Learning by Inpainting." In *CVPR*, 2016
[2] Sara Atito. "SiT: Self-Supervised VIsion Transformer." arXiv. 2021
[3] Li Dong, Songhao Piao, Furu Wei. "BEIT: BERT Pre-Training of Image Transformers." In *ICLR*, 2022
[4] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, Ross Girshick. "Masked Autoencoders Are Scalable Vision Learners." In *CVPR*, 2022
[5] Zhenda Xie, Zheng Zhang, Yue Cao, Yutong Lin, Jianmin Bao, Zhuliang Yao, Qi Dai, Han Hu. "SimMIM: A Simple Framework for Masked Image Modeling." In *CVPR*, 2022

Masking Strategies in MIM - Adversarial



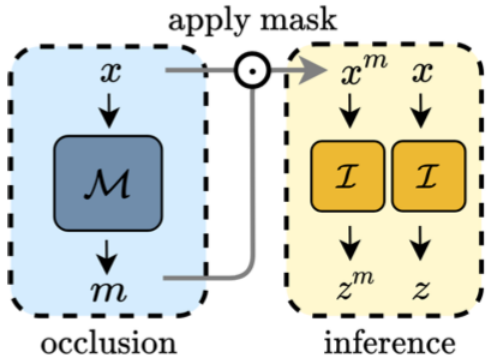


Figure 2. ADIOS Architecture.

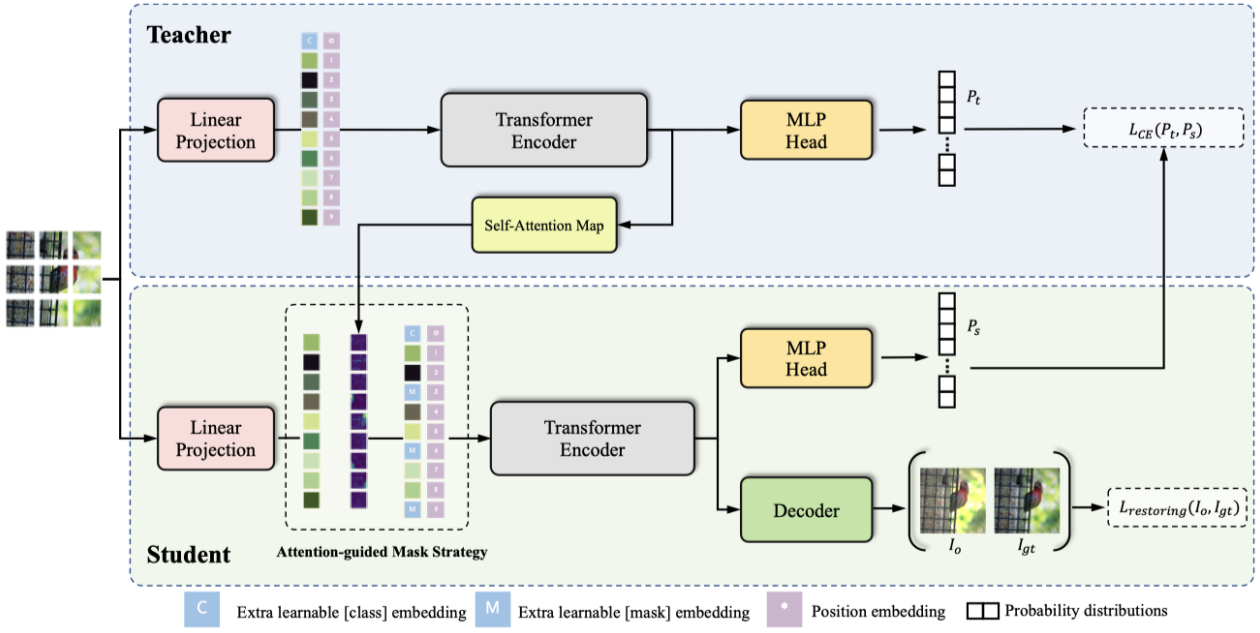
$$\mathcal{I}^\star, \mathcal{M}^\star = \arg\min_{\mathcal{I}} \max_{\mathcal{M}} \mathcal{L}(\boldsymbol{x}; \mathcal{I}, \mathcal{M}) \,. \tag{1}$$

$$\mathcal{L}_{\text{AE}}(\boldsymbol{x}; \mathcal{I}, \mathcal{M}) = \mathcal{D}(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \mathcal{D}(\boldsymbol{x}, \mathcal{I}(\boldsymbol{x} \odot \mathcal{M}(\boldsymbol{x}))), \tag{2}$$

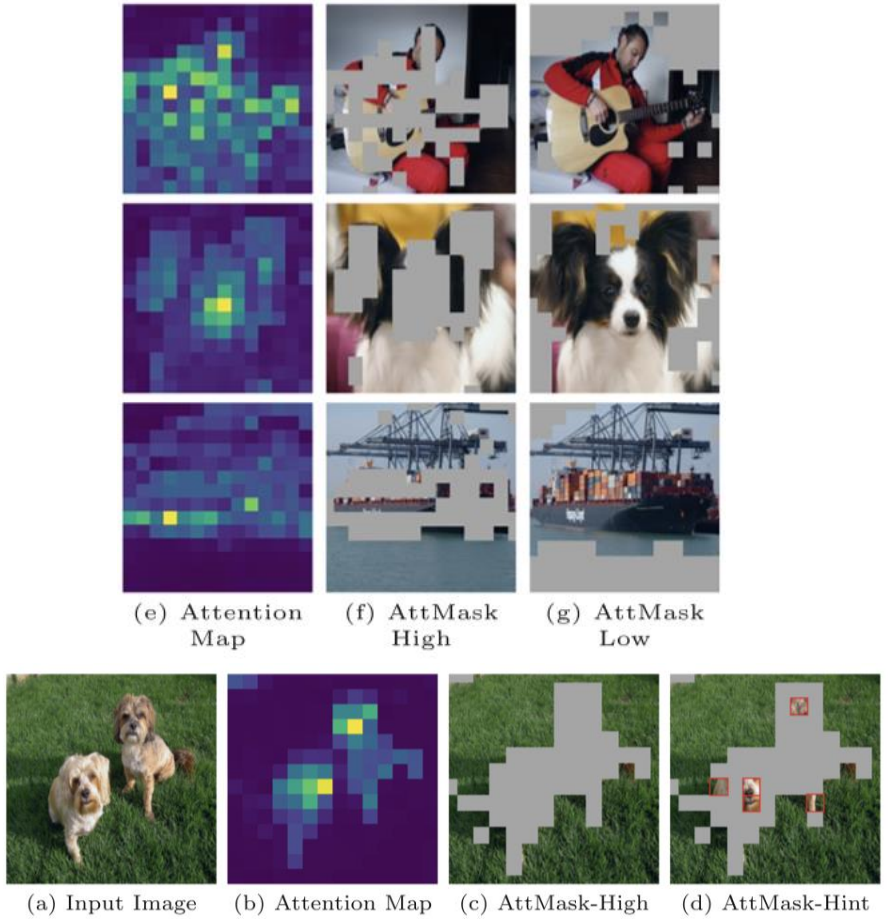
ADIOS, ICML22
“learn to mask” by adversarial training

[1] Yuge Shi, N. Siddharth, Philip Torr, Adam R. Kosiorek. “Adversarial Masking for Self-Supervised Learning.” In *ICML*. 2022

Masking Strategies in MIM - Attention



MST, NeurIPS21
Attention-guided mask

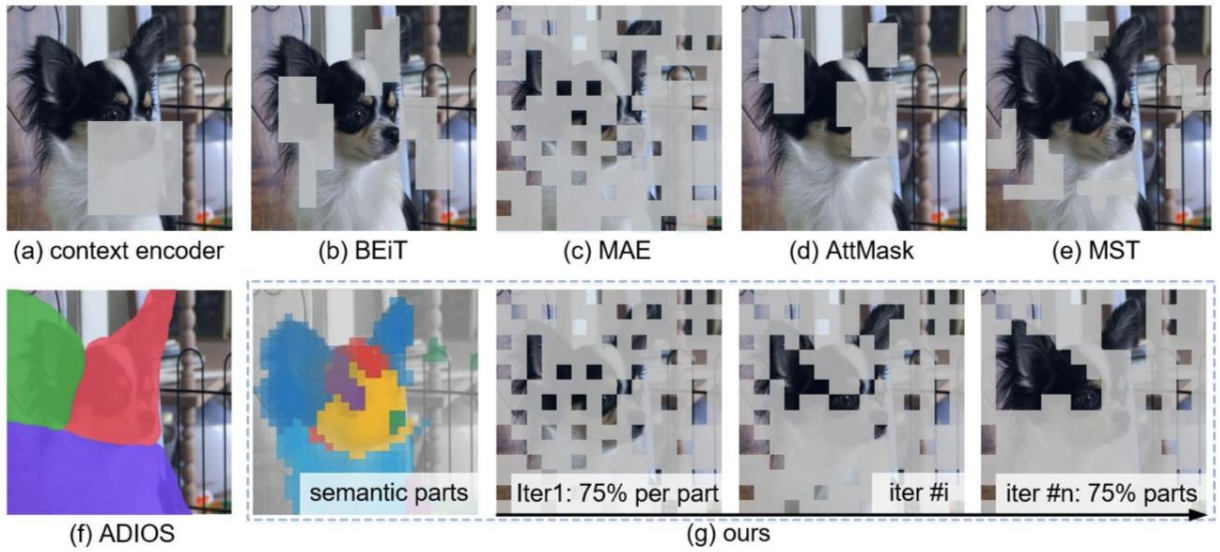


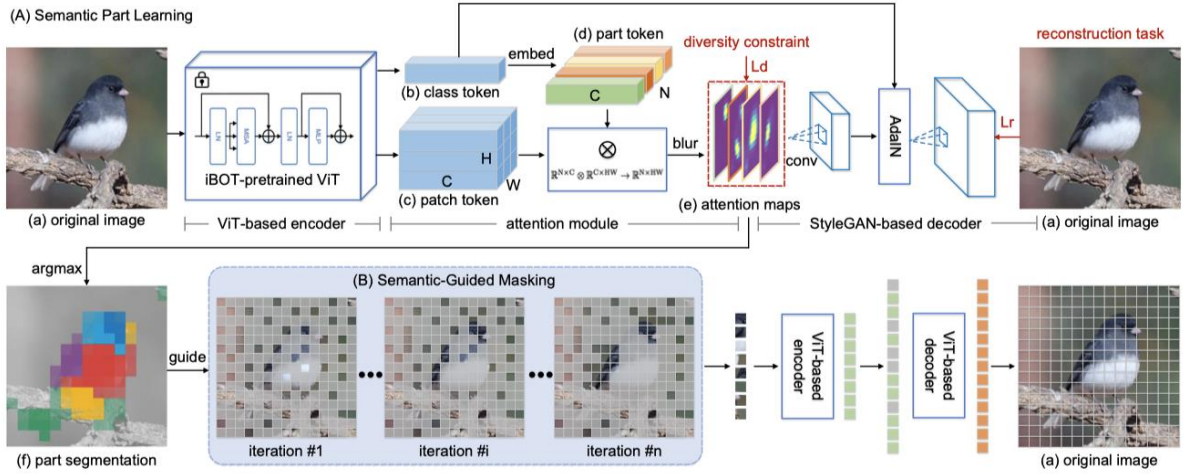
AttMask, ECCV22
Attention-guided mask

[1] Zhaowen Li, Zhiyang Chen, Fan Yang, Wei Li, Yousong Zhu, Chaoyang Zhao, Rui Deng, et al. "MST: Masked Self-Supervised Transformer for Visual Representation." In *NeurIPS*. 2021

[2] Ioannis Kakogeorgiou, Spyros Gidaris, Bill Psomas, Yannis Avrithis, Andrei Bursuc, Konstantinos Karantzalos, Nikos Komodakis. "What to Hide from Your Students: Attention-Guided Masked Image Modeling." In *ECCV*, 2022

Masking Strategies in MIM - Semantic





SemMAE, NeurIPS22

Motivation:
There is still a large gap for masked autoencoding (MAE) between vision and language due to different signal natures. A sentence can be semantically decomposed into words, while the semantic decomposition of an image is not trivial to be obtained.

Methods:
1. Identify semantic parts.
2. Dynamic easy-to-hard masking strategies:
- **a. Mask a portion of patches in each part, to learn intra-part patterns**
- **b. Mask a portion of (whole) parts in an image, to learn inter-part patterns**

[1] Gang Li, Heliang Zheng, Daqing Liu, Chaoyue Wang, Bing Su, Changwen Zheng. "SemMAE: Semantic-Guided Masking for Learning Masked Autoencoders." In *NeurIPS*. 2022

Future Directions

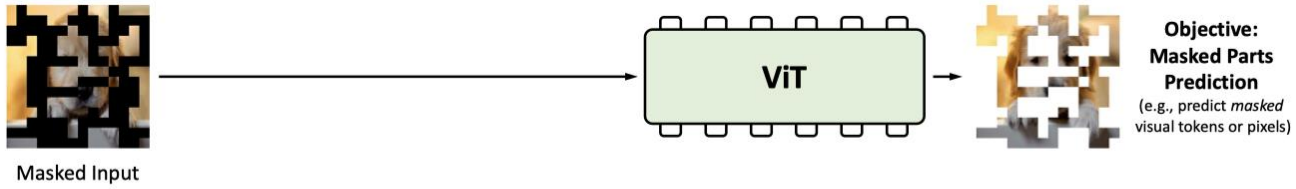
- Adversarial-based
- Attention-guided (combined with GT)
- Semantic-guided
  - What is semantic part in graph? Subgraph partition? Saliency? Prototype?

- Heuristic:
  - Structure-guided
    - Degree/Homophily-guided
    - Ego-graph-based
  - Skip-mask
  - Dynamic mask: easy-to-hard, fine-to-coarse
  - Multi-granularity

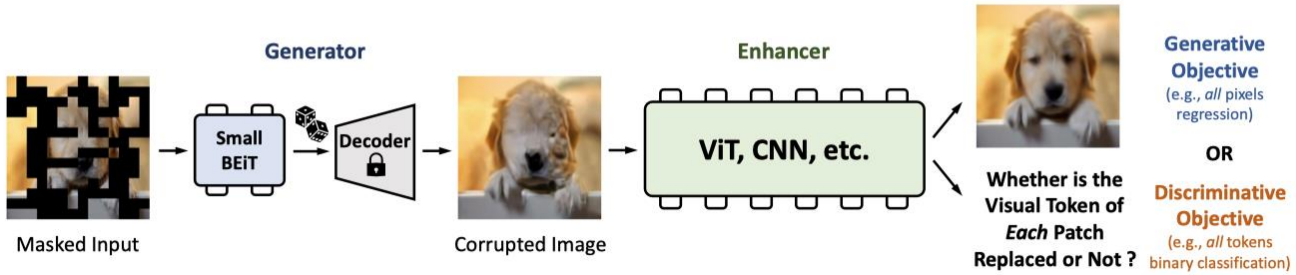
- Combine masking and augmentation

How to Mask?

CIM, ICLR, 2023



(a) Masked Image Modeling (MIM).



(b) Corrupted Image Modeling (CIM).

- **Motivation:**
  - **Deficiencies of [MASK] token:**
    - **MIM is tightly coupled with the Transformer family, and cannot combined with CNN family.**
    - **It causes a discrepancy between pre-training and fine-tuning.**
- **Methods:**
  - **Use corrupt-reconstruct to replace mask-reconstruct**
  - **Use a small trainable BEiT as an generator to corrupt the input image.**

[1] Yuxin Fang, Li Dong, Hangbo Bao, Xinggang Wang, Furu Wei. “Corrupted Image Modeling for Self-Supervised Visual Pre-Training.” In *ICLR*, 2023

CIM, ICLR, 2023

Table 1: ImageNet-1K end-to-end fine-tuning top-1 accuracy of vanilla ViT-Small/16 and ViT-Base/16 models.
†Doubled attention heads. ‡Our reproduction.

Models	PT Epochs	Top-1
ViT-Small/16 model results		
Scratch (Touvron et al., 2021a)		79.9
MoCo-v3† (Chen et al., 2021)	600	81.4
DINO (Caron et al., 2021)	1600	81.5
BEiT (Bao et al., 2021)	300	81.3
CIM-RESPIX (Ours)	300	81.5
CIM-REVDET (Ours)	300	**81.6**
ViT-Base/16 model results		
Scratch (Touvron et al., 2021a)		81.8
Scratch (He et al., 2021)		82.3
DINO (Caron et al., 2021)	1600	82.8
MoCo-v3 (Chen et al., 2021)	600	83.2
BEiT (Bao et al., 2021)	300	82.9
BEiT (Bao et al., 2021)	800	83.2
MAE‡ (He et al., 2021)	800	83.1
CIM-REVDET (Ours)	300	**83.3**
CIM-RESPIX (Ours)	300	**83.3**

[1] Yuxin Fang, Li Dong, Hangbo Bao, Xinggang Wang, Furu Wei. “Corrupted Image Modeling for Self-Supervised Visual Pre-Training.” In *ICLR*, 2023

MixMask, arXiv, 2022

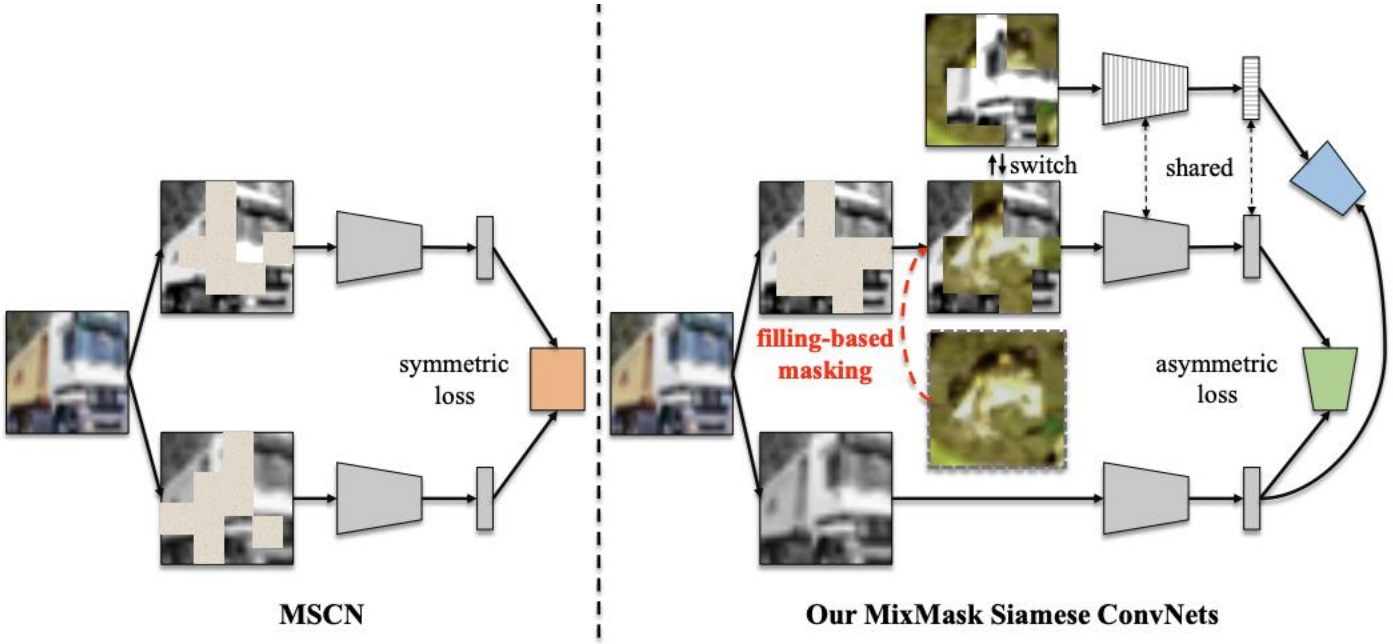


Figure 2. Illustration of the Masked Siamese ConvNets (left) and our proposed framework (right). MixMask branch incorporates asymmetry into the loss function design by generating images with different rates of similarity to the images in the original branch. In MixMask branch image of the truck is presented twice with different levels of similarity to the image in the original branch due to the regions masked with contents of another image.

[1] Kirill Vishniakov, Eric Xing, Zhiqiang Shen. “MixMask: Revisiting Masking Strategy for Siamese ConvNets.” arXiv. March 21, 2023

ModelName, Venue, Year

Wang, Yuandong, Tianyu Wo, Hongzhi Yin, Jie Xu, Hongxu Chen, and Kai Zheng. “Origin-Destination Matrix Prediction via Graph Convolution: A New Perspective of Passenger Demand Modeling.” In KDD, 2019.

ModelName, Venue, Year

Wang, Yuandong, Tianyu Wo, Hongzhi Yin, Jie Xu, Hongxu Chen, and Kai Zheng. “Origin-Destination Matrix Prediction via Graph Convolution: A New Perspective of Passenger Demand Modeling.” In KDD, 2019.